QUANTIZATION MODULATES BATCH-CONTEXT NON-DETERMINISM IN GREEDY LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantization reduces the memory and compute cost of LLM inference, and requests are usually batched to serve many of them at once. However, recent studies have shown that batching causes non-determinism in LLM inference despite the use of greedy decoding. We study whether quantization changes this batch-context non-determinism, holding prompts, decoding settings and checkpoints fixed. For that we primarily evaluate Qwen2.5-7B-Instruct and Llama-3.1-8B-Instruct across seven benchmarks and six serving contexts using BF16, GPTQ, FP8 weight-and-activation quantization, and FP8 KV-cache quantization. We measure answer disagreement across every pair of our serving contexts. FP8 weight-and-activation quantization roughly doubles how often an example changes its answer, from 8.7% to 17.2% for Qwen2.5-7B-Instruct and from 13.9% to 25.3% for Llama-3.1-8B-Instruct, while weight-only GPTQ stays close to BF16. Quantizing only the KV cache also raises the rate, and smaller evaluations on five further models show the same pattern. We trace the cause to the order in which floating-point reductions are carried out, which depends on batch composition. When we use batch-invariant kernels, the non-determinism effect vanishes for BF16 and FP8 at a throughput loss, but it remains for GPTQ.

1 INTRODUCTION

Large language models have become central infrastructure across scientific question answering, code generation, and open-ended reasoning (OpenAI et al., 2024; Rozière et al., 2024), and the pressure to serve them efficiently has grown alongside their adoption. Quantization allows serving LLMs with reduced memory and compute cost, with weight-only schemes such as GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024) reducing checkpoint size at limited accuracy cost, weight-and-activation FP8 raising arithmetic throughput (Micikevicius et al., 2022), and KV-cache quantization extending memory headroom. Open-weight models are now commonly served in a quantized form through inference engines such as vLLM (Kwon et al., 2023).

Greedy decoding at temperature zero is commonly treated as deterministic, so a fixed model and a fixed prompt set are expected to produce the same outputs under any serving configuration. However, Yuan et al. (2025) show this is not true even for BF16. Batch size, GPU count, and hardware generation all change model outputs under greedy decoding. Floating-point addition is not associative, so $(a + b) + c \neq a + (b + c)$ in finite precision, and any change in the order of operations changes the rounding (Goldberg, 1991). Under dynamic batching the numerical environment of a request is determined by the other requests it is co-scheduled with. Those requests determine the shapes that reach each kernel, and the shapes determine the order in which the reductions inside that kernel are accumulated. Consequently, a logit that rounds one way when a request is served alone can round the other way in a batch of 32, and greedy decoding propagates that difference through every subsequent token. Existing measurements of this effect vary batch size and hardware while holding the position of a request inside the queue fixed.

What remains unknown is how quantization interacts with this instability. Weight-only quantization changes every matrix multiply but leaves activations and cache at full precision. Weight-and-activation quantization additionally clips intermediate activations at each layer. KV-cache quantization degrades the stored attention state that all future tokens attend to. There is no reason to expect these

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

interventions to affect serving-context sensitivity equally, yet the quantization literature focuses almost entirely on average accuracy and efficiency.

We study this question systematically across seven benchmarks, five quantization regimes and six batched serving contexts, and then isolate the mechanism. We fix the weights, prompt text, decoding parameters and quantization recipe in every run and only vary the batch cap and the request order. Our main findings are:

- **Quantization scheme.** FP8 weight-and-activation quantization raises answer disagreement from 8.7% to 17.2% for Qwen and from 13.9% to 25.3% for Llama, and quantizing the Llama KV cache alone raises it to 18.2%. GPTQ W8A16 stores weights at the same eight bits as FP8 and remains within one point of BF16 in both models, so the nominal bit width is a poor predictor of the outcome.
- **Benchmark scores.** On GPQA Diamond a single model and regime spans up to 8.6 points of accuracy across the six contexts, with prompts and decoding held fixed. The instability concentrates on the hardest benchmarks, precisely where small score differences are used to argue for capability improvements.
- **Mechanism.** We find that a prompt served alone and the same prompt served inside a batch first select different tokens at decoding steps where the top two candidates are nearly tied, with a median margin never above 0.25 nat against 4.6 to 7.1 nats at a typical step. When we constrain kernel reduction order to be independent of batch composition, BF16 and FP8 reach exact token-sequence agreement at a throughput cost.

2 Related Work

Non-determinism and reproducibility. Floating-point non-associativity has long been recognized as a source of variability in scientific computing (Goldberg, 1991). For deep learning, Zhuang et al. (2022) identify GPU kernel non-associativity as a primary source of irreducible non-determinism, distinct from controllable stochastic sources such as dropout. For LLM inference specifically, Yuan et al. (2025) show that batch size, GPU count, and hardware generation change model outputs even under greedy decoding. On the evaluation side, Bouthillier et al. (2019) show that many performance differences fall within variance from random seed and data ordering, Biderman et al. (2026) catalogue implementation divergences that produce substantially different benchmark scores for nominally identical protocols, Madaan et al. (2024) quantify variance from benchmark subsampling, and Hochlehnert et al. (2025) find that a substantial fraction of reported reasoning progress falls within evaluation noise. We contribute a complementary source of variance, the serving context, and show that its size depends on the quantization regime.

Post-training quantization. Weight-only quantization methods preserve activations at full precision while compressing model weights: GPTQ (Frantar et al., 2023) uses layer-wise second-order information to minimize reconstruction error, and AWQ (Lin et al., 2024) selects scales based on activation magnitude. Mixed-precision decomposition (LLM.int8(), Dettmers et al. 2022) handles outlier features through a separated high-precision path. Weight-and-activation quantization additionally reduces intermediate activation precision: SmoothQuant (Xiao et al., 2023) migrates quantization difficulty from activations to weights through per-channel smoothing. FP8 inference applies this at the hardware level (Micikevicius et al., 2022). KV-cache quantization is a distinct intervention compressing stored attention states to reduce memory bandwidth over long sequences. Across all methods, standard evaluation reports average accuracy before and after quantization, leaving serving-context stability unmeasured.

High-throughput serving. PagedAttention (Kwon et al., 2023) introduced a virtual-memory abstraction for KV caches enabling continuous batching of requests with different sequence lengths, and FlashAttention (Dao et al., 2022) reorganized the attention computation to minimize HBM reads and writes. These systems batch requests dynamically, and the numerical environment of a single request changes every time the surrounding batch changes, even when the model, prompts, and decoding rule are held fixed. Our work measures how the quantization scheme modulates a model's sensitivity to that variability.

3 EXPERIMENTAL SETTING

We fix the model weights, prompt text, decoding parameters, and quantization recipe across all runs. We only vary the batch cap and request ordering to isolate the serving-context contribution from all other variance sources.

Models and benchmarks. We perform the primary analysis on two instruction-tuned models, Qwen2.5-7B-Instruct (Qwen et al., 2025) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and we evaluate them on seven benchmarks spanning a deliberate range of difficulty. MATH500 (Hendrycks et al., 2021; Lightman et al., 2024) and GPQA Diamond (Rein et al., 2024) require extended reasoning, GSM8K (Cobbe et al., 2021) and MMLU-Pro (Wang et al., 2024) involve multi-step problems, and ARC-Challenge (Clark et al., 2018), BoolQ (Clark et al., 2019) and DROP (Dua et al., 2019) are shorter-form tasks. We fixed the subsets once before any inference run and reused them unchanged for every model, regime and serving context, so subset choice cannot affect a within-subset comparison, and Appendix A records the selection rules together with the prompt construction.

Quantization regimes. We study five inference paths and two mixed-precision mitigations, which Table 5 of Appendix A lists, and we take BF16 (R0) as the full-precision reference. GPTQ W8A16 (R1) and W4A16 (R2) apply weight-only quantization calibrated on 512 sequences of general-domain prose (Frantar et al., 2023), FP8 W8A8 (R3) quantizes both weights and activations, and BF16 with an FP8 KV cache (R4) retains BF16 weights while quantizing only the cache. The mitigations M1 and M2 preserve the attention Q/K/V/O projections at BF16 within an otherwise W4A16 or FP8 W8A8 model, and we report both in Section 7. We produce the primary checkpoints with LLM Compressor v0.6.0.1 (vLLM Project, 2026a) and serve them through vLLM (Kwon et al., 2023) on NVIDIA L40S GPUs. Three configurations produced degenerate output that the answer parsers cannot read, and their exclusion is documented in Appendix L.

Serving contexts. We evaluate each model, benchmark and regime cell under six batched serving contexts, formed from two batch caps and three request orders at each cap (`b8_s1-s3` and `b32_s1-s3`), with an identical prompt set throughout. We set request order by a deterministic shuffle with a fixed seed, and because continuous batching assembles micro-batches dynamically from the queue, different orderings give each request a different co-occupancy pattern. Writing $\hat{y}_{e,c}$ for the normalized parsed answer of example e under context c, we measure answer disagreement between a pair of contexts c and c' as

$$\mathrm{Disagree}(c, c') = \frac{1}{|E_{c,c'}|} \sum_{e \in E_{c,c'}} \mathbf{1}[\hat{y}_{e,c} \neq \hat{y}_{e,c'}], \tag{1}$$

where $E_{c,c'}$ contains the examples parsed in both members of the pair. We take as our primary statistic the unweighted average of $\mathrm{Disagree}(c, c')$ over every usable (benchmark, pair) term, of which there are $7 \times 15 = 105$ when no cell is degenerate. We measure accuracy drift across the six contexts by the within-regime score range $\Delta_{\mathrm{Acc}} = \max_c \mathrm{Acc}(c) - \min_c \mathrm{Acc}(c)$. It describes disagreement over the finite set of contexts we run and assumes no deployment distribution over batch caps or request orders. Appendix C states the weights our replication design induces and reports the benchmark-equal-weighted alternative. We compare only between batched contexts, so no single-request run enters the estimator, and Appendix D reports the behaviour of repeated single-request runs measured under the same estimator.

Analysis. We compute pairwise comparisons with example-clustered bootstrap intervals over 10,000 replicates at seed 20260728, stratified by benchmark. Within each benchmark we first align the two arms onto their common example index, so an example is resampled as one unit in both arms. We exclude parse failures from Equation (1), and we get the same result under the two alternative treatments of Appendix E.

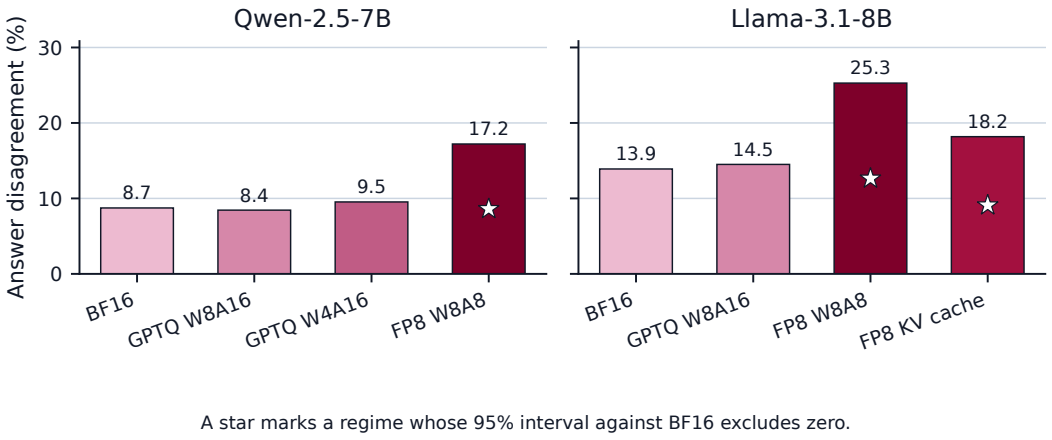


Figure 1: Answer disagreement across all 15 pairs of the six batched serving contexts, averaged over every usable (benchmark, pair) term. GPTQ remains close to BF16 at both bit widths, and FP8 roughly doubles the rate.

Table 1: Answer disagreement by quantization regime. Llama GPTQ W4A16 and Qwen BF16 with an FP8 KV cache are parse-degenerate and are excluded. The hard-subset column restricts to GPQA Diamond, MATH500 and GSM8K.

Model	Regime	Disagreement	Hard subset	Terms	95% interval vs BF16
Qwen2.5-7B	BF16	8.7%	14.6%	105/105	reference
	GPTQ W8A16	8.4%	14.0%	105/105	$[-1.00, +0.52]$
	GPTQ W4A16	9.5%	15.7%	105/105	$[-0.16, +1.46]$
	FP8 W8A8	**17.2%**	**24.0%**	105/105	$[+7.69, +9.40]$
Llama-3.1-8B	BF16	13.9%	21.1%	105/105	reference
	GPTQ W8A16	14.5%	21.4%	100/105	$[-0.81, +0.94]$
	FP8 W8A8	**25.3%**	**32.6%**	88/105	$[+7.63, +9.60]$
	BF16 with FP8 KV	18.2%	26.8%	105/105	$[+3.27, +5.09]$

4 Main Results

4.1 Disagreement by regime

Figure 1 and Table 1 report the primary result, in which answer disagreement across batched serving contexts is 8.7% under BF16 and 17.2% under FP8 W8A8 for Qwen2.5-7B-Instruct, and 13.9% and 25.3% for Llama-3.1-8B-Instruct, with weight-only GPTQ remaining close to BF16 in both models. The example-clustered bootstrap places the FP8 increase over BF16 at $+8.55$ points with a 95% interval of $[+7.69, +9.40]$ for Qwen and $+8.60$ with $[+7.63, +9.60]$ for Llama, while the GPTQ W8A16 intervals contain zero. Quantizing only the Llama KV cache raises disagreement to 18.2% while leaving weights and activations at BF16, a difference of $+4.17$ points with an interval of $[+3.27, +5.09]$.

GPTQ W8A16 and FP8 W8A8 both store weights at eight bits, and they differ in how far the reduced precision extends. Under GPTQ the quantization applies to the stored parameter tensor, while activations, cache states and attention accumulation remain in BF16. Under FP8 every intermediate value is rounded again at every layer, and Appendix B gives the precision accounting for each regime.

Accuracy responds far less to the serving context than disagreement does, since under Holm–Bonferroni correction most regimes show no significant accuracy change against BF16 even where disagreement doubles (Appendix Q). Of the answers that change across contexts, 54.3% also change the correct or incorrect label, and the direction is close to balanced at 50.2% correct-to-wrong against 49.8% wrong-to-correct, so serving-context variation adds noise to a benchmark score without imposing a systematic direction on it.

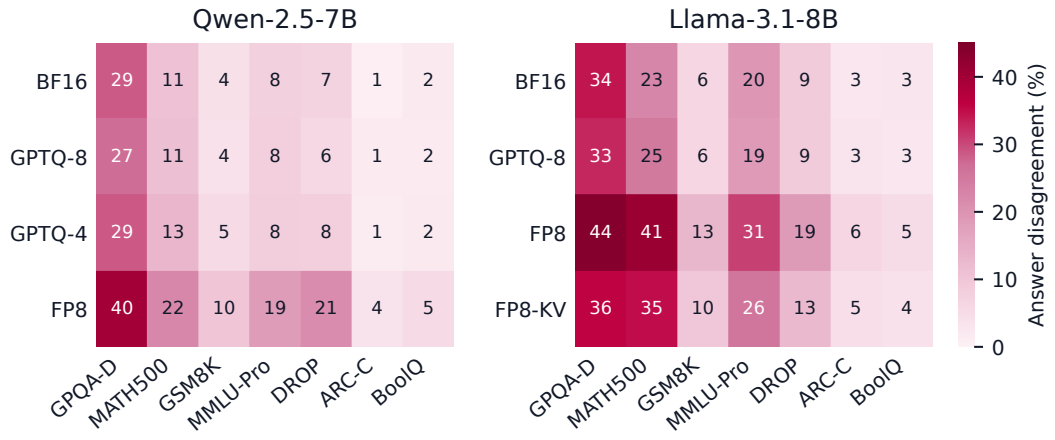


Figure 2: Per-benchmark answer disagreement for Qwen (left) and Llama (right). Rates decrease from GPQA Diamond and MATH500 to ARC-Challenge and BoolQ, following benchmark difficulty, and the FP8 row has highest disagreement in both models.

4.2 Benchmark difficulty and score drift

Disagreement is distributed very unevenly over the benchmarks (Figure 2), with GPQA Diamond reaching 28.6% for Qwen BF16 and 34.2% for Llama BF16, rising to 39.9% and 43.7% under FP8 W8A8. MATH500 reaches 11.0% and 22.6% under BF16 and 22.0% and 40.8% under FP8, while at the other end of the range Qwen ARC-Challenge records 0.9% and BoolQ 1.8%. Disagreement falls as benchmark accuracy rises, at a Spearman rank correlation of -0.80 across the model, regime and benchmark pairs (Appendix F).

These rates accumulate into drift in the reported benchmark score, and on GPQA Diamond the six contexts span 5.1 points of accuracy for Qwen BF16, 6.1 points for Llama BF16 and 8.6 points for Llama under FP8 W8A8, on a benchmark where gains of 2 to 5 points are routinely claimed. Raising the batch cap from 8 to 32 lowers accuracy in 54 of the 105 cells and raises it in 40, so no single batch size can be assumed to give the conservative estimate.

On a benchmark where a model is accurate its answer-token distribution is sharply peaked, and a perturbation of floating-point magnitude rarely moves the argmax, whereas on a benchmark where it is inaccurate the top two candidates are close on a large fraction of examples and the same perturbation suffices. We study this in Section 5.1. Figure 5 in Appendix F shows this relationship.

4.3 Token-level divergence

Answer rates are a weak indicator of how much the generated text varies, since Qwen BF16 on ARC-Challenge disagrees on under 1% of answers while 46.1% of its examples produce a different token sequence across contexts. On GPQA Diamond that fraction is 95.5% for the same model and regime and 99.0% for Llama under FP8, and across the whole matrix it ranges from 44.8% to 99.7%, with every BF16 cell included in that range. Generations take different paths in every regime, and the regimes differ in how often a different path terminates in a different answer, which makes answer disagreement a lower bound on the underlying instability. Appendix G shows the full table.

5 Mechanism

5.1 Decision margins

We run a matched GPQA Diamond experiment with per-token log probabilities recorded, and locate for each example the first token at which two serving contexts select different greedy candidates. Figure 3 reports the margin between the top two candidates at that token. The median margin is 0.125 nat in four of the six cells and never exceeds 0.25 nat in any of them, against a median of 4.6 to 7.1

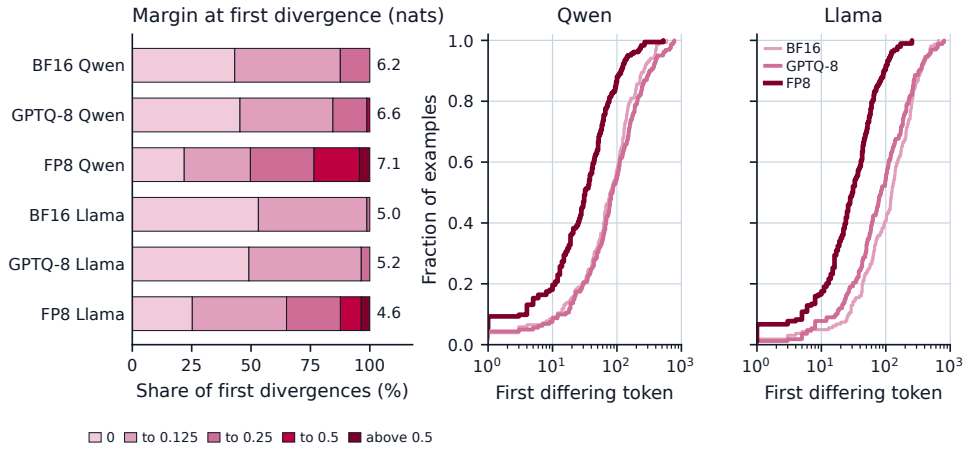


Figure 3: Left, the share of first divergences falling in each band of the top-1 minus top-2 margin, in nats, with the number beside each bar giving the median margin at a typical step of the same generation. Centre and right, the cumulative distribution of the first divergence position, on GPQA Diamond.

Table 2: Exact token-sequence matches out of 198 prompts served alone against the same prompts served inside a batch, fixed-engine comparison on GPQA Diamond.

	Qwen, L40S		Llama, L40S		Qwen, A10G	
Regime	default	invariant	default	invariant	default	invariant
BF16	45	**198**	36	**198**	45	**198**
GPTQ W8A16	37	34	31	38	36	46
FP8 W8A8	15	**198**	4	**198**	44	**198**

nats at a typical decoding step of the same generation, and in every cell at least 99% of the divergence margins fall below 1 nat. Thus, a batch-composition perturbation alters the outcome only at decoding steps that were already close to a tie.

We also locate the first divergence at a median of token 32 for Qwen and token 30.5 for Llama under FP8 W8A8, against tokens 79 and 124 under BF16. An FP8 generation therefore has most of its length remaining after the two contexts have already diverged, which is consistent with the higher answer disagreement in Table 1 and with the concentration on benchmarks whose generations are long.

5.2 Batch-invariant kernels

Batch composition determines the shapes that reach each kernel, and those shapes in turn determine the order in which partial sums are accumulated and can additionally change which kernel implementation the engine selects. To test this channel directly, we used the batch-invariant execution mode in vLLM 0.26.0 (vLLM Project, 2026b), which replaces a set of generic dense-tensor operations with implementations whose reduction order is independent of batch composition. We hold the engine configuration fixed and compare a prompt issued alone against the same prompt inside a full batch, over all 198 GPQA Diamond prompts.

Table 2 reports exact token-sequence matches under both flag states, and with the mode enabled BF16 and FP8 reach 198 exact matches out of 198 in every pair, against between 4 and 45 matches out of 198 on the default path. Constraining reduction order therefore accounts for the entire measured effect in the BF16 and FP8 regimes. A second control over 100 fixed prompts, each served alone and then inside a batch, reaches 100 matches out of 100 in every BF16 and FP8 cell under the flag against between 3 and 49 without it, on GPQA Diamond and on MATH500 alike (Appendix I).

However, the GPTQ results stay in the same range under the intervention, with the batch-invariant flag moving their counts from 37 to 34, from 31 to 38 and from 36 to 46 out of 198. We observe that the batch-invariant module overrides nine generic dense-tensor operations and contains no reference to any quantization scheme, while the GPTQ linear layer repacks its own weights and dispatches to a dedicated quantized routine that the module does not override. A determinism mode reaching only the generic paths therefore does not by itself make quantized inference reproducible. GPTQ's residual divergences under batch-invariant runs show the same near-tie signature as on the default path. As such, using batch-invariant kernels does not solve the non-determinism introduced by the GPTQ quantization regime.

Further, using batch-invariant kernels costs throughput. In a matched control at a fixed batch cap, with prefix caching disabled and a fixed generation length, BF16 throughput falls by 25.9% to 27.2% across the four cells, GPTQ by 1.8% to 9.6%, and FP8 between a 2.7% gain and a 7.1% loss. BF16 has higher costs because the replaced generic kernels perform its entire computation.

5.3 Padding controls

Batching pads co-scheduled sequences to a common length and thereby changes the shapes entering the attention computation, which makes padding amount a potential cause for non-determinism. Holding model, regime, hardware, batch cap and example count fixed, we varied whether the co-scheduled prompts were length-matched, which leaves almost no real padding, or heterogeneous under the ordinary seeded queue order.

We find that length matching leaves the effect in place, with exact token-sequence agreement between the two queues at 17.7% for Qwen BF16, 16.7% for GPTQ W8A16 and 6.6% for FP8 W8A8, and at 13.1%, 15.2% and 1.5% for the corresponding Llama comparisons. These levels are comparable to the default-path agreement in Table 2, and FP8 remains the least stable regime by a wide margin. A padding account predicts that minimising padding compresses the gap between FP8 and the weight-only regimes, and we observe no such compression. Appendix K reports answer-level rates and first-divergence positions.

Padding amount is therefore not the cause behind non-determinism. However, padding is one of several inputs that set the kernel shape, alongside co-occupancy and chunked-prefill boundaries.

5.4 The FP8 dispatch path on Ampere

We replicated the comparison on an Ampere-generation A10G, in the final pair of columns of Table 2. We observed that on the default path, FP8 records 44 exact matches out of 198 against 45 for BF16, where the same comparison on L40S records 15. Also, the median first divergence moves with it, from token 32 on L40S to token 70.5 on A10G, close to that card's BF16 value of 92.

On Ada and Hopper the attention projection weights are held as native eight-bit floating point, recorded as a `torch.float8_e4m3fn` tensor with a float32 input scale. On Ampere the same file is repacked into integer containers for a mixed-precision kernel, because the CUTLASS FP8 scaled matrix-multiply path is not compatible with Ampere architecture.

6 Other Models and GPUs

Table 3 extends the comparison to five further models, covering both dense and sparse mixture-of-experts architectures. FP8 exceeds BF16 for every model tested, and the example-clustered bootstrap interval for FP8-BF16 excludes zero in each of the models that run the FP8 W8A8 recipe and on each of their benchmarks, as Appendix M records. The Sparse MoE models show the largest gaps between FP8-BF16, while Gemma-4-E4B-it shows the smallest gap.

We find weight-only GPTQ close to BF16 on Qwen2.5-14B, as on Qwen2.5-7B. Per benchmark the 14B FP8 gap is large on GPQA Diamond and MMLU-Pro and smallest on MATH500, where the three regimes fall within 3.5 points of each other. Mistral-Nemo-12B is an exception to the GPTQ pattern, with its GPTQ leg at 19.8% above its BF16 leg at 13.8%, while every other model in our GPTQ study has GPTQ and BF16 within one point of disagreement.

Table 3: Extension studies. The four models above Gemma are evaluated on GPQA Diamond, MATH500 and MMLU-Pro under the same six-context design as the primary matrix, and Gemma on all seven benchmarks. Table 20 gives the per-benchmark values.

Model	Architecture	BF16	GPTQ	FP8	FP8 − BF16
Qwen2.5-14B-Instruct	dense	13.3%	14.3%	18.7%	+5.4 pp
Mistral-Nemo-12B	dense	13.8%	19.8%	25.6%	+11.8 pp
Qwen1.5-MoE-A2.7B-Chat	sparse MoE	27.9%	n/a	42.0%	+14.1 pp
DeepSeek-MoE-16B-chat	sparse MoE	19.3%	n/a	37.3%	+18.0 pp
Gemma-4-E4B-it	dense	8.2%	n/a	11.3%	+3.1 pp

Table 4: Median decoding throughput in tokens per second across the batched runs of the primary matrix, beside the disagreement rate of the same cell.

Model	Regime	Tokens per second	Disagreement
Qwen2.5-7B	BF16	738	8.7%
	GPTQ W8A16	976	8.4%
	FP8 W8A8	811	17.2%
Llama-3.1-8B	BF16	624	13.9%
	GPTQ W8A16	475	14.5%
	FP8 W8A8	465	25.3%

A second GPU generation. L40S and L4 are both Ada-generation cards, so replication across them does not test a change of generation, and we therefore replicated the six-context design on an NVIDIA H100 for Qwen2.5-7B-Instruct on GPQA Diamond. Disagreement is 23.9% for BF16, 27.5% for GPTQ W8A16 and 31.4% for FP8 W8A8, and the bootstrap difference for FP8 against BF16 is $+7.43$ points with a 95% interval of $[+2.85, +11.96]$, while GPTQ against BF16 is $+3.69$ with $[-0.48, +7.92]$. While the absolute level moves between Ada and Hopper, the gap between BF16 and FP8 still remains.

7 Selective Mixed Precision

We perform a preliminary mitigation, which we label M1, that preserves the attention Q/K/V/O projections in BF16 and leaves the remaining linear layers on a W4A16 recipe, and study the comparison against plain W4A16. On Qwen W4A16 disagreement falls from 9.5% to 8.6% under the all-pairs estimator of Section 3, on five of the seven benchmarks and most on the least stable of them, at a cost of 1.3 GB and about 7% of throughput. The clustered bootstrap difference is -0.75 points with a 95% interval of $[-1.60, +0.11]$. We perform the same mitigation inside an FP8 checkpoint, which we label M2, and it lowers disagreement in all three Llama benchmarks we measured, by 8 to 18 points. M2 though gives no consistent improvement for Qwen, so its benefit is model-dependent. M1 produces a much smaller effect than the batch-invariant mode and leaves disagreement well above zero, while operating on the quantized dispatch path that the batch-invariant mode leaves untouched. Appendix O provides the per-benchmark values for both mitigations.

Calibration sensitivity. We quantized three further Llama-3.1-8B checkpoints on instruction-formatted text with the chat template applied. Calibration determines whether the W4A16 checkpoint collapses at all, since the standard-calibrated checkpoint parses 0.1% of its answers and the instruction-calibrated one parses 89.5%. With the collapse gone, W4A16 records 13.2% disagreement against 13.1% for instruction-calibrated W8A16 and 23.2% for instruction-calibrated FP8 W8A8, which reproduces the same result as Table 1. The calibration corpus therefore does not explain the regime differences, and Appendix P gives the full tables.

8 DISCUSSION

Quantization choice. Batching makes greedy decoding depend on batch composition, and the quantization scheme amplifies that effect. FP8 W8A8 roughly doubles answer disagreement over BF16, while weight-only GPTQ at the same eight bits stays within one point of BF16 and runs faster than FP8 in both, as Table 4 shows. The bit width does not predict which of the two a system gets. A system choosing between the two eight-bit schemes can take GPTQ W8A16, which is the more stable and the faster of the two in both models.

Batch-invariant kernels. Constraining reduction order removes the effect for BF16 and FP8, which reach exact token-sequence agreement on every prompt we tested. BF16 gives up 25.9% to 27.2% of its throughput under the mode, because the replaced generic kernels perform its entire computation. FP8 records between a 2.7% gain and a 7.1% loss. A system already serving FP8 can have deterministic output for a few percent of throughput, and thus it's recommended to use batch-invariant kernels if FP8 quantized checkpoints are being used for LLM serving.

GPTQ mitigation. GPTQ currently has no batch-invariant solution. The batch-invariant mode replaces nine generic dense-tensor operations and reaches no quantized routine. A system that picks weight-only quantization for its stability and speed at eight bits cannot make its output reproducible with any switch the engine offers. We tried to work around this with selective mixed precision with our mitigation M1. The disagreement falls on five of the seven benchmarks, though its interval runs from -1.60 to $+0.11$. The mitigation also costs 1.3 GB and about 7% of throughput. Mixed precision is thus a lead to test at larger scale. However, we would look to the kernel first. FP8 is a quantized scheme and the batch-invariant mode already makes it deterministic for a few percent of throughput, so a batch-invariant quantized matrix multiply would put GPTQ in the same position.

9 LIMITATIONS

Absolute rates of disagreement reported in our study are specific to the hardware and serving stack. Our primary matrix covers two models of 7B to 8B, one serving framework, one quantization toolchain and two serving versions, each of which contributes to the absolute level. The extension models cover fewer benchmarks and fewer regimes, and the Mistral-Nemo GPTQ exception is one of them. The extension models are only intended to show the effect exists outside the two primary models we analyze in depth.

Both of our mixed-precision mitigations are exploratory directions. M1 covers Qwen alone and its interval contains zero, and M2 covers three benchmarks on L4 over three serving contexts. Our batch-invariant experiment holds the engine configuration fixed, which scopes that result to one configured engine, and in the padding control padding stays confounded with the kernel shape it feeds. We serve on a single GPU throughout, so tensor and context parallelism, speculative decoding, multiple-token prediction and chunked prefill are untested, and the kernel-path dependence of Section 5.2 gives reason to expect they influence the effect. The seven benchmarks are English-language datasets evaluated under greedy decoding at one prompt template, and thus the effect in multi-lingual setting under varied prompt configurations is not tested by us.

10 CONCLUSION

We measured how the quantization scheme changes batch-context non-determinism in greedy decoding. FP8 weight-and-activation quantization raises answer disagreement from 8.7% to 17.2% for Qwen2.5-7B-Instruct and from 13.9% to 25.3% for Llama-3.1-8B-Instruct, factors of 1.97 and 1.82, while weight-only GPTQ at the same nominal bit width stays within one point of BF16 in both. Divergences begin at decoding steps where the top two candidates are nearly tied, and the FP8 excess over BF16 disappears on a GPU that never selects the native FP8 path. Constraining reduction order to be independent of batch composition removes the non-determinism effect for BF16 and FP8, at a throughput cost, while the GPTQ results stay near their default counts under the same mode.

Ethics Statement

In this work, we evaluate publicly available model checkpoints on publicly available English-language benchmarks. Our study involves no human subjects, no personal data and no new data collection. One benchmark, GPQA Diamond, is access-gated and requires authentication, and we accessed it under its stated terms. Our results describe a reproducibility property of quantized serving and we see no direct path from them to harmful application. The main societal consideration we note is that undisclosed serving configuration can make published benchmark comparisons less informative than they appear.

Reproducibility Statement

Section 3 specifies the models, benchmark subsets, quantization regimes, serving contexts, decoding policy and answer parsers. Equation (1) and the surrounding text define the estimator, and Appendix C states the weights the replication design induces along with the benchmark-equal-weighted alternative. Appendix A records the subset selection rules including the seeds, so the evaluation sets can be rebuilt. Appendix E reports two alternative parse-failure treatments, Appendix L lists the configurations excluded for degenerate output, and Appendix R records the software environment separately for each experiment. We provide these details for reproducibility of the results.

AI Use Statement

In this work, we used generative AI tools for help with the writing of the paper. We used generative AI tools for writing polishing, paraphrasing, and grammar correction in all the sections of the paper. The generative AI tools were not used to draft the whole paper or sections, but each section after being drafted by us, was fed to generative AI tools for rephrasing and better readability. We have used generative AI for help in coding the execution of the study. Each code and experiment was manually verified and debugged by us for correctness. We have not used generative AI tools for ideation, and experimentation plan for the study. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, Charles Lovering, Niklas Muennighoff, Ellie Pavlick, Jason Phang, Aviya Skowron, Samson Tan, Xiangru Tang, Kevin A. Wang, Genta Indra Winata, François Yvon, and Andy Zou. Lessons from the trenches on reproducible evaluation of language models, 2026. URL https://arxiv.org/abs/2405.14782.

Xavier Bouthillier, César Laurent, and Pascal Vincent. Unreproducible research is reproducible. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 725–734. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/bouthillier19a.html.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359. Curran Associates, Inc., 2022. doi: 10.52202/068431-1189. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30318–30332. Curran Associates, Inc., 2022. doi: 10.52202/068431-2198. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c3ba4962c05c49636d4c6206a97e9c8a-Paper-Conference.pdf.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246/.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023. URL https://arxiv.org/abs/2210.17323.

David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.*, 23(1):5–48, March 1991. ISSN 0360-0300. doi: 10.1145/103162.103163. URL https://doi.org/10.1145/103162.103163.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong,

Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria

Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=90UrTTxp5O.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In P. Gibbons, G. Pekhimenko, and C. De Sa (eds.), *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf.

Lovish Madaan, Aaditya K. Singh, Rylan Schaeffer, Andrew Poulton, Sanmi Koyejo, Pontus Stenetorp, Sharan Narang, and Dieuwke Hupkes. Quantifying variance in evaluation benchmarks, 2024. URL https://arxiv.org/abs/2406.10229.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. Fp8 formats for deep learning, 2022. URL https://arxiv.org/abs/2209.05433.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike

Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL https://arxiv.org/abs/2308.12950.

vLLM Project. LLM compressor. https://docs.vllm.ai/en/stable/features/quantization/llm_compressor/, 2026a. Accessed 2026-05-05.

vLLM Project. Batch invariance. https://docs.vllm.ai/en/latest/features/batch_invariance/, 2026b. Accessed 2026-05-05.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290. Curran Associates, Inc., 2024. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ad236edc564f3e3156e1b2feafb99a24-Paper-Datasets_and_Benchmarks_Track.pdf.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and mitigating numerical sources of nondeterminism in llm inference. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 169819–169851. Curran Associates, Inc., 2025. doi: 10.52202/085713-5653. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f80094a824ba5912d4a2de169c404a40-Paper-Conference.pdf.

Donglin Zhuang, Xingyao Zhang, Shuaiwen Song, and Sara Hooker. Randomness in neural network training: Characterizing the impact of tooling. In D. Marculescu, Y. Chi, and C. Wu (eds.), *Proceedings of Machine Learning and Systems*, volume 4, pp. 316–336, 2022. URL https://proceedings.mlsys.org/paper_files/paper/2022/file/427e0e886ebf87538afdf0badb805b7f-Paper.pdf.

A Experimental Protocol

Benchmark subsets. MMLU-Pro uses 100 examples from each of 14 categories after a category-wise shuffle with seed 20260503, giving 1,400 examples. DROP uses the first 1,200 validation examples after a shuffle with the same seed. MATH500, GSM8K, GPQA Diamond, ARC-Challenge and BoolQ use their full evaluation splits, at 500, 1,319, 198, 1,172 and 3,270 examples respectively. GPQA Diamond is used from the gated `gpqa_diamond` huggingface repo and requires authentication to download, and its per-example answer options are shuffled deterministically with a per-example seed derived from the example index.

Table 5: Inference regimes studied. R0 to R4 make up the main matrix and M1 and M2 are the two selective mixed-precision constructions of Section 7.

ID	Regime	Description
R0	BF16	Full-precision reference.
R1	GPTQ W8A16	Weight-only, 8-bit, activations in BF16.
R2	GPTQ W4A16	Weight-only, 4-bit, activations in BF16.
R3	FP8 W8A8	FP8 weights and activations, BF16 KV cache.
R4	BF16 with FP8 KV	BF16 weights and activations, FP8 KV cache.
M1	W4A16 with BF16 attention	W4A16 except attention Q/K/V/O projections remain BF16.
M2	FP8 W8A8 with BF16 attention	FP8 W8A8 except attention Q/K/V/O projections remain BF16.

Prompts. We build a single prompt file before any inference run and hold it fixed across all runs. Each prompt is one user turn holding the question text and a final-answer instruction that makes extraction independent of reasoning chain length, and we wrap that turn with the chat template the

checkpoint ships, so the system message is whatever the model defines. Appendix A.1 gives the templates.

Decoding. Decoding is greedy in every run, with temperature 0 and top-p equal to 1, and the maximum generation length varies by benchmark at 1,024 tokens for MATH500, GPQA Diamond and GSM8K, 512 for MMLU-Pro, ARC-Challenge and DROP, and 256 for BoolQ.

Serving contexts. The six batched contexts are `b8_s1`, `b8_s2`, `b8_s3`, `b32_s1`, `b32_s2` and `b32_s3`, where the number is the batch cap and the suffix is the queue seed. We submit requests in a deterministic shuffled order fixed by the seed, and vLLM assembles micro-batches from the resulting queue.

A.1 Prompt templates

The user turn is the question body followed by a blank line and one of two final-answer instructions. MATH500, GSM8K and DROP take the free-form instruction, and GPQA Diamond, ARC-Challenge, MMLU-Pro and BoolQ take the lettered one.

ONE_LINE_ANSWER (free-form)

```
End your response with exactly one line in this format:
Final answer: <answer>
```

ONE_LINE_MCQ (multiple-choice)

```
End your response with exactly one line in this format:
Final answer: <A/B/C/D/...>
```

Braces mark the fields filled per example, and `{choices}` is the option list with one lettered option per line. For GPQA Diamond the four options are shuffled per example before the letters are assigned, under a seed derived from the example index, and for MMLU-Pro and ARC-Challenge the letters follow the order the source dataset gives.

MATH500

```
Problem: {problem}
{ONE_LINE_ANSWER}
```

GSM8K

```
Question: {question}
{ONE_LINE_ANSWER}
```

GPQA Diamond

```
Question: {question}
{choices}
{ONE_LINE_MCQ}
```

ARC-Challenge

```
Question: {question}
{choices}
{ONE_LINE_MCQ}
```

MMLU-Pro

```
Question: {question}
{choices}
{ONE_LINE_MCQ}
```

DROP

```
Passage: {passage}
Question: {question}
{ONE_LINE_ANSWER}
```

BoolQ

```
Passage: {passage}
Question: {question}
A. Yes
B. No
{ONE_LINE_MCQ}
```

B Precision Details

We produce every regime with llmcompressor, targeting `Linear` modules with `lm_head` excluded, so embeddings, normalization layers and the LM head stay in BF16 in all of them. Quantization scales are themselves unquantized, so the effective bits per weight exceed the nominal figure slightly. Q/K/V/O projections are targeted on the same terms as every other linear layer, which means FP8 W8A8 gives them FP8 E4M3 weights with FP8 E4M3 inputs and GPTQ W8A16 gives them INT8 weights with BF16 inputs, and M1 and M2 are the two mitigations in the study that exempt them. Table 6 gives this for every regime.

Table 6: Where reduced precision appears in each regime. GPTQ W8A16 and FP8 W8A8 store weights at the same eight bits and differ in numeric format and in whether the layer inputs are reduced as well.

Regime	Targeted linear weights	Linear inputs	KV cache
R0 BF16	BF16	BF16	BF16
R1 GPTQ W8A16	INT8	BF16	BF16
R2 GPTQ W4A16	INT4	BF16	BF16
R3 FP8 W8A8	FP8 E4M3	FP8 E4M3	BF16
R4 BF16 with FP8 KV	BF16	BF16	FP8
M1	INT4, Q/K/V/O in BF16	BF16	BF16
M2	FP8 E4M3, Q/K/V/O in BF16	FP8 E4M3, Q/K/V/O in BF16	BF16

C The All-Pairs Estimate

The six batched contexts are three request orders at each of two batch caps, from which the 15 unordered pairs consist of three within-cap comparisons at the smaller cap, three within-cap comparisons at the larger cap, and nine comparisons across caps. Weighting all pairs equally places weight 0.2 on each within-cap category and 0.6 on the across-cap category. With k request orders per cap the across-cap share is $k/(2k-1)$, so it is 0.6 for our $k = 3$ and approaches 0.5 as more orders are added, which makes the weights a consequence of the replication design.

Table 7 gives the three categories separately, and the FP8 minus BF16 gap for Qwen is between $+6.9$ and $+10.0$ points depending on the category, and the Llama gap is between $+9.0$ and $+11.8$ points, with the same ordering of the regimes in each category.

Averaging conventions. Our primary metric is a flat average over every usable (benchmark, pair) term, and a benchmark-equal-weighted average, taking the mean of seven per-benchmark means, is identical whenever every benchmark contributes all 15 pairs, which holds for all Qwen cells and for Llama BF16 and Llama with an FP8 KV cache. The two differ for Llama GPTQ W8A16 (100 of 105 terms, 14.5% flat against 14.0% benchmark-weighted) and Llama FP8 W8A8 (88 of 105 terms, 25.3% flat against 22.6% benchmark-weighted). This difference arises because BoolQ contributes only 3 of its 15 possible pairs in the Llama FP8 cell and BoolQ's rate is low, so equal

Table 7: Answer disagreement split by the kind of context pair. Figure 4 shows the same values.

Model	Regime	Within cap 8	Within cap 32	Across caps
Qwen2.5-7B	BF16	8.2%	8.0%	9.2%
	GPTQ W8A16	8.1%	8.8%	8.5%
	GPTQ W4A16	8.5%	8.5%	10.2%
	FP8 W8A8	15.1%	17.9%	17.7%
Llama-3.1-8B	BF16	12.8%	13.9%	14.3%
	GPTQ W8A16	13.4%	13.5%	14.3%
	FP8 W8A8	21.8%	25.1%	26.1%
	BF16 with FP8 KV	17.3%	17.7%	18.6%

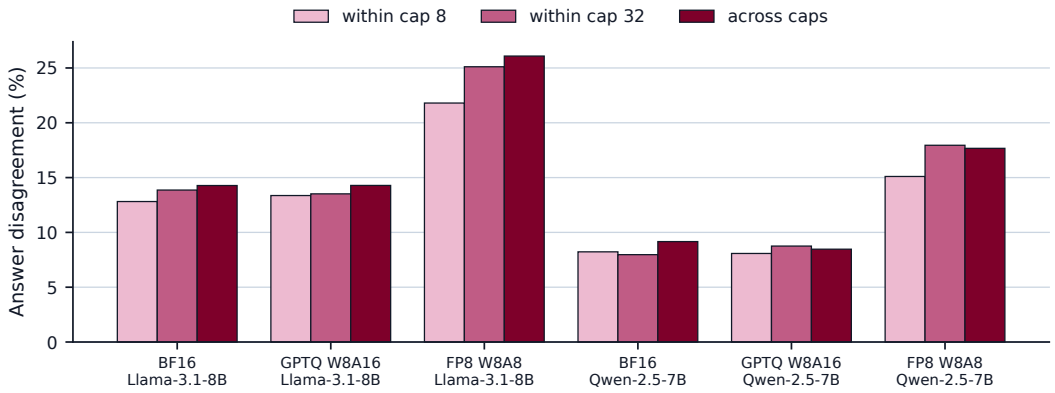


Figure 4: Answer disagreement by kind of context pair. Rates rise slightly with batch cap and are generally highest across caps, and the regime ordering is identical in all three categories.

benchmark weighting gives that low rate a full one-seventh share. The FP8 against BF16 ordering and its significance hold under either convention. The bootstrap intervals in Table 1 are computed on the benchmark-equal-weighted basis over examples common to both arms, which is the more conservative of the two for the Llama FP8 comparison.

D Repeated Single-Request Runs

A single-request run is an obvious reference point for batched runs. However, we do not use one, because an analysis built on one such run cannot separate serving-context sensitivity from what that run itself contributes. To show this, we ran the single-request configuration three times across all seven benchmarks, for four Qwen regimes and for three Llama regimes, so the three repeats form a fourth category beside the two within-cap categories and the across-cap category.

Table 8: Qwen2.5-7B-Instruct answer disagreement (%) by category of context pair, over all seven benchmarks. Every category is an all-pairs average within its own set of runs, so no run acts as a reference.

Regime	Within b=1	Within cap 8	Within cap 32	Across caps
BF16	8.6	8.2	8.0	9.2
GPTQ W8A16	10.1	8.1	8.8	8.5
GPTQ W4A16	4.3	8.5	8.5	10.2
FP8 W8A8	12.9	15.1	17.9	17.7

Repeated single-request runs disagree with each other about as much as batched contexts do, and for FP8 W8A8 by rather less, which taken at face value would place most of the variation inside a configuration that never changes. Table 9 splits the within b=1 column into its three constituent pairs. FP8 quantization still amplifies the disagreement within single-request runs as well.

Table 9: The same within b=1 figure split into its three constituent pairs, where r_2 and r_3 were run in one later campaign and r_1 belongs to the primary matrix.

Regime	r_1 vs r_2	r_1 vs r_3	r_2 vs r_3
BF16	13.0	13.0	0.0
GPTQ W8A16	15.1	15.1	0.0
GPTQ W4A16	6.5	6.5	0.0
FP8 W8A8	19.3	19.3	0.0

The two repeats from the later campaign reproduce each other exactly, at 0.0% in all four Qwen regimes, and each of them differs from the original run by 6.5 to 19.3 points.

Table 10 repeats the design for Llama over the same seven benchmarks and its three primary regimes. BF16 and GPTQ W8A16 behave as Qwen does, with the later pair at 0.0% and each of those two repeats differing from the original run by 15.2 points under BF16 and by 21.6 under GPTQ W8A16, while FP8 W8A8 does not. Its two later repeats disagree on 11.1% of answers.

A single-request run is thus not a dependable reference in either model. Where the repeats within a campaign do agree exactly, the configuration is stable inside that campaign, so the single-request instability is carried by whatever separates the two campaigns, and we cannot identify what that is. Where they do not agree, as under Llama FP8 W8A8, the configuration is not reproducible even at one request, so it cannot serve as the fixed point a reference-based estimator needs.

Table 10: Llama-3.1-8B answer disagreement (%) by category of context pair, over all seven benchmarks, on the same design as Table 8. The three right-hand columns split the within b=1 figure into its constituent pairs.

	Category				Within b=1 pairs		
Regime	Within b=1	Within cap 8	Within cap 32	Across caps	r_1 vs r_2	r_1 vs r_3	r_2 vs r_3
BF16	10.2	12.8	13.9	14.3	15.2	15.2	0.0
GPTQ W8A16	14.4	13.4	14.7	14.9	21.6	21.6	0.0
FP8 W8A8	20.2	21.8	26.0	26.5	24.3	25.1	**11.1**

The three batched columns of Table 10 are a partition of the fifteen pairs the primary estimator averages, into three within cap 8, three within cap 32 and nine across caps, so recombining them in that proportion returns the Llama rates of Table 1 to within 0.2 points.

E Answer Extraction

We extract answers by a benchmark-specific rule. For multiple-choice benchmarks the parser reads the letter or word on the final-answer line, and for MATH500 and GSM8K it takes the final numerical or algebraic expression, including a LaTeX boxed expression when the model omits the final-answer line. We normalise extracted strings by collapsing whitespace, lower-casing and stripping trailing punctuation before comparison, and we count parse failures separately and exclude them from Equation (1).

Table 11 reports three treatments of unreadable answers on the same design. Counting a pair as a disagreement whenever either side failed to parse raises every rate, most in the Llama cells where failures are more common, and restricting to examples that parsed in all six contexts lowers every rate slightly, with absolute levels moving by up to 13 points between the three columns. The FP8 gap over BF16 stays open in all of them and the ordering of the three regimes is identical, so the result holds under every treatment of unreadable answers.

F Per-Benchmark Breakdown

Table 12 gives every cell of the primary matrix, and Figure 5 plots the same rates against benchmark accuracy.

Table 11: Answer disagreement (%) under three treatments of unreadable answers, on the same all-pairs batched design. Excluding failures is the treatment used throughout the paper.

Model	Regime	Exclude failures	Failure counts as disagreement	Complete cases only
Qwen2.5-7B	BF16	8.7	10.4	8.6
	GPTQ W8A16	8.4	10.1	8.1
	FP8 W8A8	17.2	19.7	16.5
Llama-3.1-8B	BF16	13.9	21.8	12.4
	GPTQ W8A16	14.5	22.6	13.1
	FP8 W8A8	25.3	35.6	22.3

Table 12: Per-benchmark answer disagreement (%). Regime codes follow Table 5. Parenthesised counts give usable pairs out of 15 where fewer than 15 are available.

Model	Regime	GPQA-D	MATH500	GSM8K	MMLU-Pro	DROP	ARC-C	BoolQ
Qwen2.5-7B	R0	28.6	11.0	4.1	7.8	7.0	0.9	1.8
	R1	26.8	11.4	3.8	7.9	6.1	1.5	1.7
	R2	28.7	13.3	5.1	8.4	8.1	1.3	1.9
	R3	39.9	22.0	10.0	18.5	21.3	3.5	5.3
Llama-3.1-8B	R0	34.2	22.6	6.4	19.5	8.9	2.7	2.9
	R1	32.8	24.9	6.4	18.9	8.8	3.1	2.7 (10)
	R3	43.7	40.8	13.2	30.7	18.7 (10)	6.5	4.9 (3)
	R4	35.7	34.9	9.7	25.6	12.6	4.6	4.0

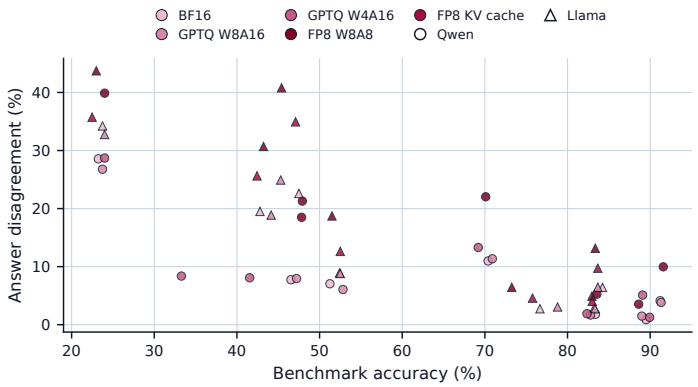


Figure 5: Benchmark accuracy against answer disagreement across every benchmark and regime cell. Harder tasks occupy the high-disagreement corner regardless of regime, and FP8 markers appear above BF16 markers within each benchmark cluster.

G Token-Level Divergence

Table 13 reports the fraction of examples whose generated token sequence differs across at least two of the six batched contexts and Figure 6 plots it against answer disagreement.

H First-Divergence Margins

Table 14 gives the values behind Figure 3, where the divergence margin is the difference between the top-1 and top-2 log-probabilities at the first token at which the isolated and batched runs differ, and the typical margin is the median of that difference over all other decoding steps of the same generation. We measure all three regimes under default execution, since BF16 and FP8 have no divergences left once batch-invariant kernels are enabled.

Table 13: Token-sequence divergence (%), the fraction of examples producing more than one distinct token sequence across the six batched contexts. Regime codes follow Table 5.

Model	Regime	GPQA-D	MATH500	GSM8K	MMLU-Pro	DROP	ARC-C	BoolQ
Qwen2.5-7B	R0	95.5	77.4	65.0	64.2	47.2	46.1	50.2
	R1	96.0	78.8	65.1	62.7	49.7	45.2	48.6
	R2	90.9	78.0	58.5	44.8	49.2	47.3	53.4
	R3	94.9	98.4	95.7	94.9	91.6	70.4	94.3
Llama-3.1-8B	R0	96.5	87.2	72.3	93.6	67.8	66.6	80.2
	R1	96.0	87.0	78.5	93.8	69.6	66.2	76.8
	R3	99.0	99.4	98.0	99.7	95.8	93.3	94.3
	R4	99.0	96.2	90.8	98.4	82.4	81.1	90.7

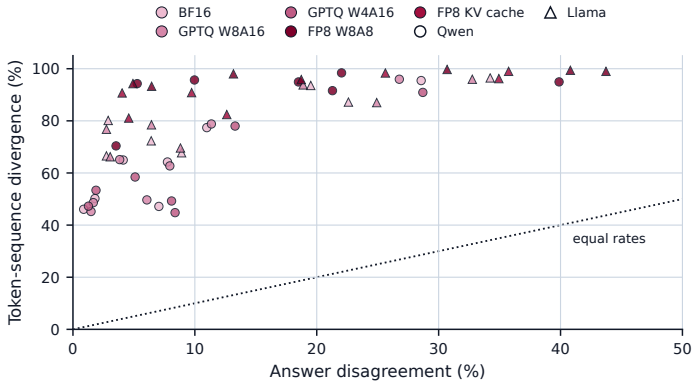


Figure 6: Token-sequence divergence against answer disagreement. Every point falls well above the diagonal, so sequences branch far more often than the final answer changes.

Table 14: First-divergence analysis on GPQA Diamond, 198 prompts per cell, under default execution, comparing each prompt served alone against the same prompt inside a batch of 32, where the diverging column counts the examples whose two runs differ at any token.

Model	GPU	Regime	Diverging	Median token	Margin	Typical	< 1 nat
Qwen2.5-7B	L40S	BF16	153	79.0	0.125	6.19	100.0%
		GPTQ W8A16	161	85.0	0.125	6.63	100.0%
		FP8 W8A8	183	**32.0**	0.250	7.13	100.0%
Llama-3.1-8B	L40S	BF16	162	124.0	0.000	5.00	100.0%
		GPTQ W8A16	167	84.0	0.125	5.25	100.0%
		FP8 W8A8	194	**30.5**	0.125	4.63	99.0%
Qwen2.5-7B	A10G	BF16	153	92.0	0.125	6.31	100.0%
		GPTQ W8A16	162	79.0	0.125	6.41	100.0%
		FP8 W8A8	154	70.5	0.125	6.50	100.0%

The A10G block is the same comparison on hardware where the native FP8 path is not selected. The FP8 median first divergence moves from token 32 on L40S to 70.5 on A10G, close to that card's BF16 value of 92, which matches the exact-match result in Table 2. Figure 7 plots those counts.

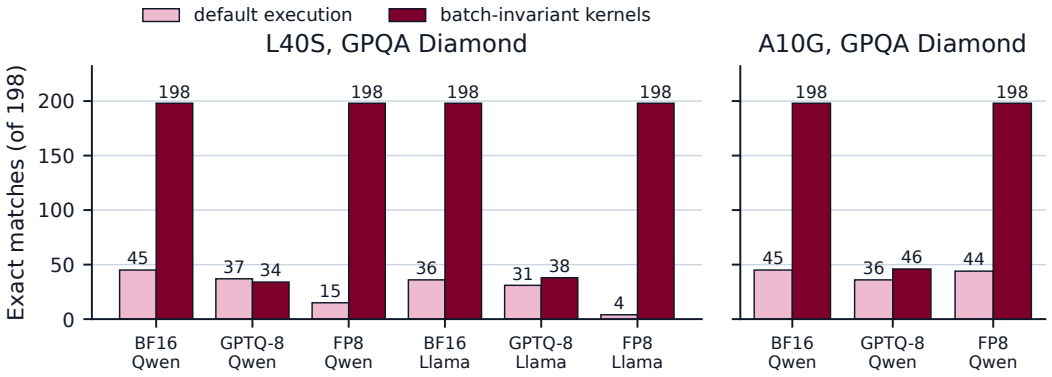


Figure 7: Exact token-sequence matches over 198 GPQA Diamond prompts in the fixed-engine comparison of an isolated call against a batched call. BF16 and FP8 reach complete agreement once reduction order is constrained, and the GPTQ path stays in the same range. On A10G, where the native FP8 path is never selected, FP8 under default execution records 44 matches against 45 for BF16.

I Batch Invariance on a Second Benchmark

Table 2 is measured on GPQA Diamond. We perform the same batch-invariant experiment on MATH500, over 100 fixed prompts per cell on L40S under vLLM 0.26.0. Table 15 gives both benchmarks. BF16 and FP8 W8A8 reach 100 exact token-sequence matches out of 100 on both benchmarks and in both models once reduction order is constrained, from as few as 3 by default, while GPTQ W8A16 moves by at most 5 of 100 in either direction in all four of its cells.

MATH500 starts from a higher default level than GPQA Diamond in five of the six cells, at 49 against 23 for Qwen under BF16 and 44 against 24 under GPTQ W8A16, which follows the difficulty relation of Appendix F, since a benchmark on which a model is more accurate diverges less often.

Table 15: Exact token-sequence matches out of 100 fixed prompts served alone against the same prompts served inside a batch.

		GPQA Diamond		MATH500	
Model	Regime	default	invariant	default	invariant
Qwen2.5-7B	BF16	23	**100**	49	**100**
	GPTQ W8A16	24	19	44	41
	FP8 W8A8	9	**100**	20	**100**
Llama-3.1-8B	BF16	20	**100**	30	**100**
	GPTQ W8A16	13	15	36	32
	FP8 W8A8	3	**100**	3	**100**

J Throughput Cost

At the same nominal eight bits, GPTQ W8A16 is the faster of the two quantized regimes in both models, by 20% for Qwen and by 2% for Llama, and it is the more stable of the two by about nine and eleven points, as Table 4 records.

Table 16 reports the throughput lost when batch-invariant kernels are enabled on the same engine configuration. The control is matched across the two flag states, using the same 100 prompts, the

same batch cap of 32, prefix caching disabled and a fixed generation length, so the two arms differ only in the kernels that execute, and Figure 8 shows the same values.

Table 16: Percentage of throughput lost when batch-invariant kernels are enabled, at a matched batch cap of 32. A negative value is a small gain.

Model	Benchmark	Regime	Throughput lost
Qwen2.5-7B	GPQA Diamond	BF16	26.8%
		GPTQ W8A16	2.8%
		FP8 W8A8	4.6%
Qwen2.5-7B	MATH500	BF16	26.1%
		GPTQ W8A16	9.6%
		FP8 W8A8	1.5%
Llama-3.1-8B	GPQA Diamond	BF16	25.9%
		GPTQ W8A16	1.8%
		FP8 W8A8	7.1%
Llama-3.1-8B	MATH500	BF16	27.2%
		GPTQ W8A16	2.9%
		FP8 W8A8	−2.7%

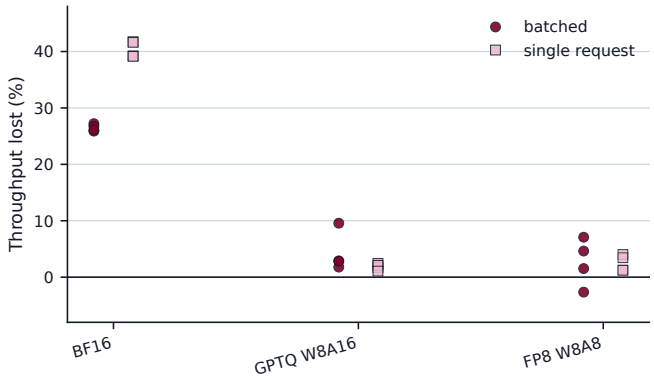


Figure 8: Throughput lost to batch invariance, one marker per model and benchmark. Circles are the batched measurements of Table 16 and squares repeat them at a single request, which is the setting that isolates kernel cost from batching.

K Padding Controls

We test whether padding amount is a separate driver for the non-determinism, with four tests, all of them on GPQA Diamond with 198 prompts per cell and with model, regime, hardware, batch cap and example count held fixed, varying only whether the queue is sorted by rendered prompt length. Sorting raises the Spearman correlation between queue position and prompt token length from -0.08 for Qwen and -0.07 for Llama under the ordinary seeded order to $+1.00$ in both.

Table 18 reports the answer-level and token-level comparison between the length-matched queue and the ordinary one. Agreement stays low in every cell, and the regime ordering is preserved, with FP8 the least stable in both models. If padding drives the effect, the first divergence should arrive later once the queue is length-matched, and instead it arrives earlier in five of six cells. Padding also predicts that a shorter prompt diverges more often, since it is co-scheduled with longer ones and receives more shape padding, and that this relationship weakens once the queue is length-matched. Table 17 reports the correlation between prompt length and divergence.

Five of the six cells are indistinguishable from zero and their signs do not agree, at $+0.011$ to $+0.090$ in four of them. Qwen under FP8 W8A8 records -0.196 with $p = 0.006$, and it holds at -0.195

after prompt length and divergence are both adjusted for generation length. Length matching drops it to -0.046, so this one regime follows the prediction and the other five do not.

Table 17: Spearman correlation between prompt token length and whether an individual example diverges from its single-request run, GPQA Diamond, 198 prompts per cell. If padding drives divergence, this value is negative under the ordinary queue and weaker under length matching.

		Ordinary queue		Length-matched		
Model	Regime	ρ	p	ρ	p	Partial ρ
Qwen2.5-7B	BF16	$+0.055$	0.439	$+0.160$	0.024	$+0.045$
	GPTQ W8A16	$+0.090$	0.206	$+0.034$	0.632	$+0.067$
	FP8 W8A8	-0.196	**0.006**	-0.046	0.518	-0.195
Llama-3.1-8B	BF16	$+0.011$	0.881	$+0.022$	0.753	$+0.029$
	GPTQ W8A16	-0.007	0.925	-0.038	0.593	$+0.001$
	FP8 W8A8	$+0.027$	0.708	$+0.035$	0.627	$+0.045$

Table 18: Length-matched queue against the ordinary seeded queue, GPQA Diamond, 198 prompts. Valid counts the examples parsed in both arms.

Model	Regime	Answer disagreement	Exact token match	Valid
Qwen2.5-7B	BF16	29.1%	17.7%	189
	GPTQ W8A16	27.1%	16.7%	188
	FP8 W8A8	37.6%	6.6%	181
Llama-3.1-8B	BF16	34.8%	13.1%	115
	GPTQ W8A16	29.2%	15.2%	120
	FP8 W8A8	42.0%	1.5%	112

The fixed-engine comparison behind Table 2 runs with CUDA graphs enabled in both flag states, so shape padding to captured graph sizes is present in both arms while BF16 and FP8 still reach exact agreement. Within the present control the queue composition is what varies, and padding amount moves together with the kernel shape each request meets.

L Collapsed Configurations

In our study, three configurations produced repetitive output that the answer parsers could not read, with parse success near zero and accuracy near zero. These are Llama GPTQ W4A16, Llama with attention-preserved W4A16, and Qwen BF16 with an FP8 KV cache. We exclude them from the disagreement analysis, because a rate computed over a handful of usable examples is not comparable to a rate computed over a full benchmark. Two further Llama cells are partially affected, with GPTQ W8A16 losing 5 of 105 pairs and FP8 W8A8 losing 17 of 105, concentrated in BoolQ and DROP, and Table 1 prints the usable term count for every row.

M Extension Studies

Figure 9 shows the extension results and Figure 10 shows the H100 replication. Table 20 breaks the four models that run the full design down by benchmark and gives the bootstrap interval on the FP8 against BF16 difference in each cell, and every interval excludes zero. The Mistral-Nemo GPTQ is the only checkpoint in the study where a weight-only regime exceeds its own FP8 leg, at 12.7% against 7.3% on GPQA Diamond with an interval of $[-9.4, -1.6]$, while on the other two benchmarks FP8 is the higher of the pair.

Because the reported rate drops a pair whenever either side is unreadable, the pool of examples behind it differs from pair to pair. Table 19 recomputes the three low-parse models on one fixed pool per cell, keeping only examples readable in all six contexts, which is the convention Appendix E applies to the primary matrix. No cell moves up, the largest move down is 5.6 points, and no ordering within

a benchmark changes. FP8 stands above BF16 in all nine rows, and the Mistral GPQA Diamond inversion survives at 12.1% against 7.3%.

Table 19: The three extension models with the lowest parse coverage, under the per-pair rule used throughout and under a complete-case rule that fixes one pool per cell.

		BF16		GPTQ		FP8		
Model	Benchmark	Per pair	Fixed	Per pair	Fixed	Per pair	Fixed	n fixed
Qwen1.5-MoE-A2.7B	GPQA Diamond	5.9	5.3	n/a		15.0	13.6	187–189
	MATH500	48.8	45.6	n/a		72.6	70.6	411–420
	MMLU-Pro	29.0	24.6	n/a		38.5	32.9	775–943
Mistral-Nemo-12B	GPQA Diamond	3.0	3.0	12.7	12.1	7.3	7.3	192–198
	MATH500	28.4	27.4	30.0	28.2	52.0	49.0	415–459
	MMLU-Pro	9.9	9.2	16.6	15.8	17.6	16.6	1,201–1,289
DeepSeek-MoE-16B	GPQA Diamond	10.3	9.0	n/a		21.4	19.6	122–138
	MATH500	27.8	25.2	n/a		53.3	50.6	310–368
	MMLU-Pro	19.8	14.5	n/a		37.3	32.8	567–660

Table 20: Per-benchmark answer disagreement for the four extension models that run the full six-context design, with the example-clustered bootstrap interval on the FP8 minus BF16 difference in percentage points.

Model	Benchmark	BF16	GPTQ	FP8	FP8 − BF16
Qwen2.5-14B	GPQA Diamond	20.1%	21.5%	27.3%	+7.2 [+3.3, +11.1]
	MATH500	8.1%	9.7%	11.6%	+3.5 [+2.0, +5.0]
	MMLU-Pro	11.7%	11.6%	17.2%	+5.5 [+4.9, +8.6]
Qwen1.5-MoE-A2.7B	GPQA Diamond	5.9%	n/a	15.0%	+9.1 [+5.7, +12.3]
	MATH500	48.8%	n/a	72.6%	+23.8 [+20.6, +26.9]
	MMLU-Pro	29.0%	n/a	38.5%	+9.5 [+7.8, +11.2]
Mistral-Nemo-12B	GPQA Diamond	3.0%	12.7%	7.3%	+4.3 [+1.7, +6.9]
	MATH500	28.4%	30.0%	52.0%	+23.6 [+20.3, +26.9]
	MMLU-Pro	9.9%	16.6%	17.6%	+7.7 [+6.2, +9.1]
DeepSeek-MoE-16B	GPQA Diamond	10.3%	n/a	21.4%	+11.0 [+6.6, +15.3]
	MATH500	27.8%	n/a	53.3%	+25.6 [+22.1, +29.0]
	MMLU-Pro	19.8%	n/a	37.3%	+17.5 [+15.2, +19.8]

N CHECKPOINTS

The primary Qwen and Llama checkpoints for every regime were produced by us with a single quantization toolchain from pinned model revisions, and the extension models draw on a mix of checkpoints we produced and externally published ones. The Qwen2.5-14B extension uses `Qwen/Qwen2.5-14B-Instruct`, `Qwen/Qwen2.5-14B-Instruct-GPTQ-Int8` and `RedHatAI/Qwen2.5-14B-Instruct-FP8-dynamic`. Qwen1.5-MoE-A2.7B-Chat has no GPTQ leg because the serving framework does not support the required quantized mixture-of-experts path at the version used, and DeepSeek-MoE-16B-chat has no GPTQ leg because no such published checkpoint exists. Table 21 reads the recipe out of each GPTQ checkpoint's own `quantization_config`.

The toolchain also decides which kernel serves the checkpoint, and our engine log names `AllSparkLinearKernel` for the two primary models and `MarlinLinearKernel` for the Qwen2.5-14B extension, so the batch-invariant result of Section 5.2 is a statement about the AllSpark path. The batch-invariant module overrides no quantized operation. We did not inspect the AllSpark source itself, so the coverage gap is a property of that override list. All three checkpoints are standard eight-bit weight-only GPTQ.

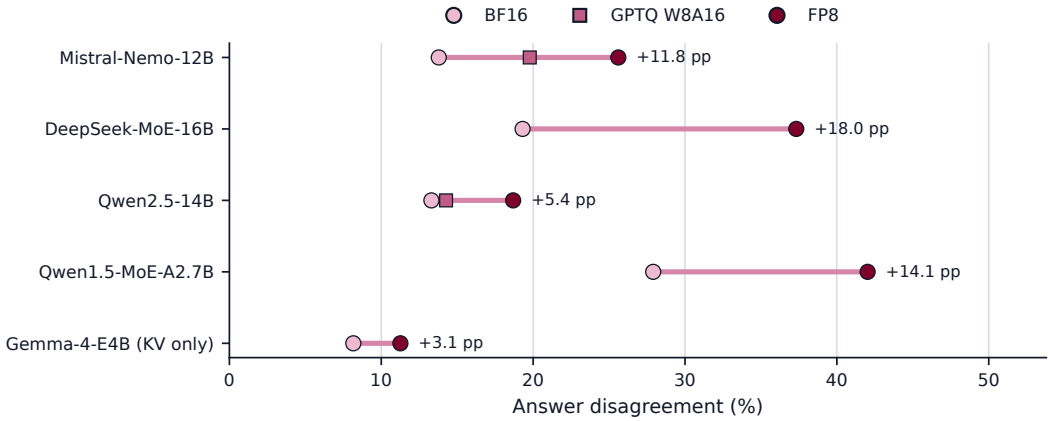


Figure 9: Answer disagreement in the extension studies. Squares mark GPTQ W8A16 where a checkpoint was available.

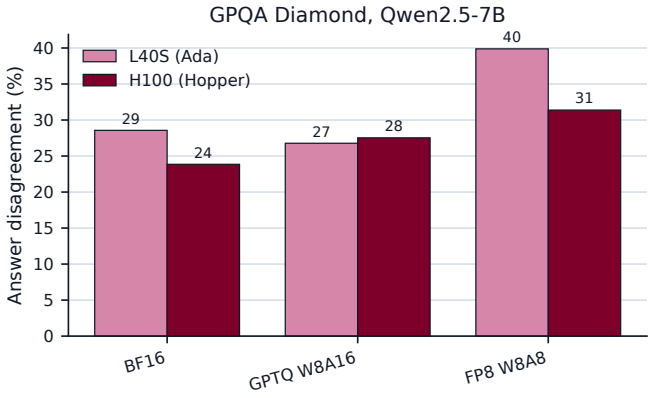


Figure 10: GPQA Diamond disagreement for Qwen2.5-7B on two GPU generations, each under its own serving stack.

Table 21: What the label GPTQ W8A16 refers to in each part of the study, read from the stored `quantization_config` of each checkpoint. The two models served by the external toolchain are the two whose GPTQ leg departs from the pattern in the primary matrix.

Checkpoints	Toolchain	Scale granularity	Calibration set
Qwen2.5-7B, Llama-3.1-8B (ours)	`llmcompressor`	per-channel, no grouping	documented
Qwen2.5-14B-Instruct-GPTQ-Int8	AutoGPTQ	group-wise, group size 128	not recorded
Mistral-Nemo GPTQ-INT8	AutoGPTQ	group-wise, group size 128	not recorded

The two primary quantized regimes differ in scale granularity as well as in numeric format. GPTQ W8A16 stores integer weights with per-output-channel scaling and no grouping, and leaves activations and the KV cache in BF16. FP8 W8A8 stores eight-bit floating-point weights and activations with per-tensor static scaling and also leaves the KV cache in BF16. Granularity could therefore explain the FP8 result on its own. We rule that out with the Ampere comparison of Section 5.4, which loads the same checkpoint on both cards, so granularity is identical while the instability disappears on one of them.

O MITIGATION DETAILS

Qwen2.5-7B-Instruct uses grouped-query attention with 28 query heads and 4 key-value heads, so the preserved Q/K/V/O projections are a smaller share of the checkpoint than the head count alone suggests, which holds the size cost to 1.3 GB. Table 22 reports per-benchmark values for the attention-preserved W4A16 configuration on Qwen, where the largest reduction falls on GPQA Diamond, the least stable of the seven benchmarks, and the two most stable benchmarks move by less than a quarter of a point. We ran the configuration in all six batched contexts on all seven benchmarks, so its rates use the same 105 terms as the primary matrix and its W4A16 column reproduces Table 12. Checkpoint size grows from 4.3 GB to 5.6 GB and throughput falls from 117 to 109 tokens per second.

We applied the same intervention inside an FP8 checkpoint, on NVIDIA L4 under vLLM 0.26.0 over three batched contexts and three benchmarks, and Table 23 sets it against the BF16 and FP8 legs of that campaign. For Llama the attention-preserved rate is below the FP8 rate on all three benchmarks and within two points of the BF16 rate in each of them. For Qwen it is one point below the FP8 rate on GPQA Diamond and marginally above it on the other two, so it remains close to FP8 throughout. Preserving the attention projections recovers the BF16 level for one model while not having much effect for the other one.

Table 22: Attention-preserved mixed precision against the W4A16 baseline on Qwen2.5-7B, under the flat all-pairs estimator of Table 12 over the same six batched contexts, so the W4A16 column is the Qwen W4A16 row of that table. The difference column is taken before rounding.

Benchmark	W4A16	Attention preserved	Difference
GPQA Diamond	28.7%	25.8%	−2.9 pp
MATH500	13.3%	11.7%	−1.6 pp
GSM8K	5.1%	4.2%	−0.9 pp
MMLU-Pro	8.4%	8.5%	+0.1 pp
ARC-Challenge	1.3%	1.5%	+0.2 pp
DROP	8.1%	6.5%	−1.6 pp
BoolQ	1.9%	1.8%	−0.1 pp
Flat all-pairs mean	9.5%	8.6%	−1.0 pp

Table 23: Attention-preserved FP8 against the BF16 and FP8 legs of the same L4 campaign, over three batched contexts.

Model	Benchmark	BF16	FP8	Attention preserved	Change
Llama-3.1-8B	GPQA Diamond	31.8%	40.7%	32.4%	−8.3 pp
	MATH500	22.4%	38.5%	20.5%	−18.0 pp
	MMLU-Pro	13.1%	23.6%	12.8%	−10.8 pp
Qwen2.5-7B	GPQA Diamond	28.1%	40.8%	39.8%	−1.0 pp
	MATH500	6.6%	12.0%	14.1%	+2.1 pp
	MMLU-Pro	8.0%	16.3%	16.4%	+0.1 pp

The same campaign covered a fourth benchmark, DROP, under BF16 and FP8 W8A8 with no attention-preserved leg, so it falls outside Table 23. Llama moves from 9.3% to 18.0% and Qwen from 6.3% to 20.1%, on about 3,000 usable pairs per cell.

P INSTRUCTION-CALIBRATED GPTQ

We quantized three additional Llama-3.1-8B regimes with the calibration corpus replaced by instruction-formatted text under the chat template, covering W8A16, W4A16 and FP8 W8A8. These runs cover batched contexts only, and their rates use the same flat all-pairs estimator as the primary matrix, with all 105 terms available in each of the three cells against 100 and 88 for the standard W8A16 and FP8 legs. Table 24 gives the six rates together with parse success and accuracy on the effective-correct basis, where an unreadable answer counts as incorrect.

Calibration decides whether the W4A16 checkpoint collapses at all, since parse success rises from 0.1% to 89.5% and effective accuracy from 0.0% to 66.4%. Once collapse is prevented, W4A16 records 13.2% cross-context disagreement, which places it beside instruction-calibrated W8A16 at 13.1% and well below instruction-calibrated FP8 W8A8 at 23.2%. The calibration change moves W8A16 by -1.4 points and FP8 W8A8 by -2.1 points, and the separation between the weight-only and the weight-and-activation regimes is about ten points under either corpus.

Table 24: Llama-3.1-8B under the standard calibration corpus and under instruction-formatted calibration, on the six batched contexts.

Regime	Calibration	Disagreement	Terms	Parse success	Accuracy
GPTQ W8A16	standard	14.5%	100/105	92.1%	68.3%
	instruction	13.1%	105/105	91.5%	68.7%
GPTQ W4A16	standard	n/a	39/105	0.1%	0.0%
	instruction	13.2%	105/105	89.5%	66.4%
FP8 W8A8	standard	25.3%	88/105	90.4%	65.1%
	instruction	23.2%	105/105	91.7%	68.2%

Q STATISTICAL TABLES

Bootstrap intervals use 10,000 example-clustered replicates with seed 20260728, stratified by benchmark, with the two arms aligned onto their common example index within each benchmark before resampling. Table 1 reports the intervals for the primary matrix and Section 6 reports them for the H100 replication. The FP8 against BF16 interval is $[+10.36, +13.30]$ for Mistral-Nemo and $[+16.01, +20.05]$ for DeepSeek-MoE, each stratified over the three benchmarks those models cover, and Table 20 gives every per-benchmark interval. The Gemma row of Table 3 has no interval, since the only quantized regime available for that model changes the KV cache, which is a different intervention from the weight and activation recipe the other rows use. Restricting Gemma to GPQA Diamond, MATH500 and GSM8K, the three hard benchmarks, raises both of its rates while preserving their order, to 12.4% under BF16 and 16.2% with the FP8 KV cache.

Accuracy comparisons use McNemar tests on paired per-example outcomes with Holm–Bonferroni correction across the regime and benchmark pairs. Most regimes show no significant accuracy change relative to BF16 even where disagreement doubles.

R SOFTWARE ENVIRONMENT

Our experiments span more than one serving stack. The primary matrix contains 742,755 rows recorded under vLLM 0.9.2 with PyTorch 2.7.0 on CUDA 12.6 and 354,708 rows under vLLM 0.20.1 with PyTorch 2.11.0 on CUDA 13.0, all on NVIDIA L40S. The batch-invariant experiments, the margin analysis, the padding controls, the extension models, the H100 replication and the A10G comparison use vLLM 0.26.0 with PyTorch 2.11.0. The attention-preserved FP8 cells in Appendix O were run separately on NVIDIA L4 under vLLM 0.26.0.