LSMXQ: LAYERWISE SALIENCE-AWARE MIXED-PRECISION QUANTIZATION FOR EFFICIENT LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training quantization (PTQ) is an effective approach for reducing the memory footprint of large language models (LLMs), enabling larger models to fit within limited GPU memory and improving inference efficiency under memory-bound workloads. While recent sub-4-bit quantization methods can achieve high accuracy compared to FP16 baselines, they do not necessarily leverage modern hardware-supported FP4 Tensor Core architectures, limiting their ability to fully utilize the computational throughput of modern GPUs, particularly at larger batch sizes. In this work, we introduce LSMXQ, a hardware-aware, mixed-precision, post-training quantization approach that allocates low-bit representations under an average bit budget while minimizing layer-wise propagated quantization error. LSMXQ is designed for coalesced memory access and efficient NVFP4 Tensor Core execution, supporting both memory-bound low-batch inference and high-throughput large-batch workloads. Across a broad evaluation of LLMs and zero-shot benchmarks, LSMXQ retains up to 98.5% of FP16 accuracy at 2-bit precision, while matching FP16 accuracy at 2.5- and 3-bit precision. For inference efficiency, LSMXQ achieves up to 3.71× the throughput of FP16 in batch-1 inference, while delivering substantial throughput improvements at larger batch sizes in vLLM.

1 INTRODUCTION

LLMs have reached scales where the cost of storing and moving model weights has become a fundamental constraint on inference efficiency. For individual users, a large weight footprint can prevent a model from fitting into the available GPU memory, limiting the models that can be deployed locally. In production environments, the same footprint increases GPU memory consumption and may require additional GPUs to serve a given workload, while leaving less memory available for runtime state, including the KV cache, activations, and other execution buffers. This reduced memory headroom can ultimately constrain serving capacity and the number of concurrent requests that can be supported. Low-bit post-training quantization (PTQ) offers a practical way to reduce this memory footprint and the associated memory traffic without requiring retraining of the underlying model (Frantar et al., 2022; Lin et al., 2024; Chee et al., 2023; Tseng et al., 2024a;b).

However, ultra-low-bit quantization introduces a fundamental trade-off between quantization granularity, precision allocation, and representation overhead. Weight groups do not contribute equally to quantization error, motivating finer-grained grouping so that precision can be allocated according to the sensitivity of individual groups (Huang et al., 2025). At the same time, reducing the group size indefinitely is not practical, as finer-grained representations require more metadata and scaling information per weight, increasing the storage and decoding overhead. Moreover, assigning a uniform bitwidth to all groups constrains the weights within each group to the same limited set of quantization levels, regardless of their individual sensitivity or value distribution. At very low bitwidths, this restriction can unnecessarily reduce the representational diversity within a group, forcing heterogeneous weights toward the same quantization levels and increasing quantization error (Chee et al., 2023; Tseng et al., 2024a;b). These observations suggest that effective ultra-low-bit quantization must jointly consider the granularity of weight groups, the precision assigned to each group, and the overhead introduced by the resulting representation.

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.

LSMXQ uses fine-grained, salience-aware groups to enable flexible bitwidth allocation while controlling representation overhead. We use 32-weight groups as a practical granularity that increases allocation flexibility compared with larger groups while keeping metadata overhead manageable and providing a CUDA-friendly basis for subsequent warp-level execution and weight repacking. For each group, we evaluate 1-, 2-, 3-, and 4-bit quantization together with candidate per-16-weight E4M3 FP8 and per-256-weight FP16 scaling factors using a Hessian-weighted SSE objective (Frantar & Alistarh, 2022; Frantar et al., 2022). For the 1-, 2-, and 3-bit configurations, a subset search evaluates the possible subsets for each candidate representation and selects the subset yielding the lowest perplexity. The resulting per-group, per-bit costs are then jointly optimized with dynamic programming under a target average bitwidth constraint, selecting the combination that minimizes aggregate weighted reconstruction error rather than assigning bits solely from independent group sensitivity rankings.

For batch-1 inference, LSMXQ can directly use its native scale representation. For high-batch serving in vLLM (Kwon et al., 2023), however, we convert the scale representation to the format expected by an optimized NVFP4 GEMM path (NVIDIA, 2025). This adapts the original per-16-weight E4M3 and per-256-weight FP16 scaling factors to the existing low-precision kernel interface, allowing LSMXQ to reuse the optimized NVFP4 execution path without requiring a separate kernel.

The central hypothesis of the paper is that effective ultra-low-bit inference requires the quantizer and the execution representation to be designed together. The experiments therefore evaluate LSMXQ along four dimensions: quantization accuracy, effective memory compression, kernel-level execution efficiency, and end-to-end serving performance.

Our contributions are as follows:

- **Hardware-aware mixed-bit quantization.** We introduce LSMXQ, a hardware-aware mixed-bit quantization framework designed around efficient NVFP4 execution. Rather than optimizing compression independently of execution, LSMXQ constrains the quantized representation to remain compatible with the scaling structure and memory layout required by efficient low-bit GPU kernels.

- **Salience-aware bitwidth allocation.** We develop a salience-aware quantization procedure that combines Hessian-derived sensitivity with group-level reconstruction error. For 1-, 2-, and 3-bit representations, we evaluate candidate E2M1 level subsets on the calibration set and select the subset yielding the lowest perplexity for each target average bitwidth. Given the selected subset, dynamic programming determines the bitwidth assignment across groups under a prescribed average bitwidth constraint.

- **NVFP4-compatible execution.** We develop an NVFP4-compatible representation and conversion path that enables the resulting mixed-bit weights to leverage optimized NVFP4 GEMM kernels. The representation supports native execution for batch-1 inference, while for high-batch serving, the scale representation is converted to the layout expected by the NVFP4 execution path. This provides a practical way to integrate LSMXQ with existing low-precision serving infrastructure while retaining flexibility for specialized execution kernels.

2 Related Work

2.1 Post-Training Quantization for Large Language Models

Post-training quantization methods seek to reduce model precision without retraining the full model. Reconstruction-based approaches such as GPTQ (Frantar et al., 2022) use second-order information to compensate for quantization error, while activation-aware approaches such as AWQ (Lin et al., 2024) identify important weights or channels using activation statistics. Other work has explored curvature-aware, information-theoretic, and codebook-based approaches for ultra-low-bit settings (Chee et al., 2023; Tseng et al., 2024a;b).

2.2 Mixed-Bit and Ultra-Low-Bit Quantization

SliM-LLM (Huang et al., 2025) is a closely related salience-driven mixed-precision quantization approach that assigns different bitwidths to structured weight groups. However, its low-bit representations are primarily designed around conventional low-precision inference pipelines rather than a native NVFP4-compatible execution representation. Furthermore, while existing non-uniform approaches often rely on post hoc thresholding or ranking heuristics, LSMXQ formulates precision allocation as an explicit discrete optimization problem subject to an exact average bitwidth budget. In contrast to purely algorithmic approaches, LSMXQ jointly optimizes the quantized representation alongside its target GPU execution path, enabling the resulting mixed-bit weights to directly leverage optimized low-precision GEMM kernels in high-batch serving.

2.3 Low-Precision GPU Execution

Modern GPUs increasingly provide specialized execution paths for ultra-low-precision matrix operations, with FP4-family formats such as NVFP4 and MXFP4 enabling native 4-bit computation on supported hardware (Rouhani et al., 2023). These formats are not defined by the low-bit values alone: efficient execution depends on the compatibility between the compressed representation, scaling metadata, memory layout, and kernel interface. NVFP4 is particularly relevant to LSMXQ because its hierarchical scaling structure and native low-precision matrix-multiply path provide a practical execution target for sub-4-bit inference on supported NVIDIA GPUs (NVIDIA, 2025).

Prior quantization systems have demonstrated that co-designing the compressed format with the kernel is critical for realizing actual hardware speedups. For example, Marlin (Frantar et al., 2025) provides a highly optimized 4-bit GPTQ execution path that parallelizes loading, dequantization, and computation, showing that memory layout and kernel interface substantially dictate practical inference throughput. LSMXQ extends this hardware-aware perspective to mixed-bit sub-4-bit quantization by targeting the execution structure of NVFP4.

3 LSMXQ

For each specified target average bitwidth, LSMXQ allows the subset of E2M1 quantization levels used by the corresponding 1-, 2-, and 3-bit representations to be configured. We empirically evaluate candidate subsets on the calibration set and select the subset that yields the lowest perplexity for the target bitwidth. Given the selected subset, each quantization group is then evaluated at candidate bitwidths using scale-optimized Hessian-weighted reconstruction error. Dynamic programming then selects the bitwidth assignment that minimizes the aggregate weighted reconstruction error while satisfying the specified target average bitwidth and preserving a CUDA-friendly, coalesced memory layout. The optimization is performed separately for each specified target average bitwidth, with the corresponding DP formulation and problem size determined by that target configuration and its coalesced packing constraints. Once the bitwidth assignment is determined, the resulting representation is directly used by the native LSMXQ execution path, or converted to the layout and scaling format expected by optimized NVFP4 kernels during high-batch serving.

3.1 LSMXQ Quantization and NVFP4-Compatible Representation

LSMXQ assigns 1-, 2-, 3-, or 4-bit weight representations under a target average bitwidth constraint while maintaining an execution representation compatible with efficient low-precision GPU kernels. Rather than applying a uniform bitwidth to all weight groups, LSMXQ allocates different bitwidths according to their quantization sensitivity. Following GPTQ (Frantar et al., 2022), we estimate the local sensitivity of input column j as

$$s_j = \frac{1}{(H^{-1})_{jj}},$$

where H is the Hessian of the reconstruction objective estimated from the calibration activations. Layers are quantized sequentially, and the calibration inputs to each layer are obtained from the

already-quantized preceding layers, so that the Hessian and reconstruction costs account for quantization error propagated from earlier layers. We normalize these sensitivities within each GPTQ column block of size $B = 128$,

$$\tilde{s}_j = \frac{s_j}{\frac{1}{B}\sum_{k\in\mathcal{B}} s_k},$$

where $\mathcal{B}$ denotes the current column block. The blocks are processed sequentially following the standard second-order block update scheme. LSMXQ then uses this Hessian-derived sensitivity together with the group-level reconstruction error to determine the bitwidth assigned to each quantization group through dynamic programming under hardware-aware memory constraints. The resulting representation is designed to account for both the statistical properties of the weights and the requirements of efficient execution. In particular, LSMXQ can map sub-4-bit weight representations to an NVFP4-based execution path, which is natively supported and highly optimized on NVIDIA Blackwell GPUs, while the underlying quantization strategy is not intrinsically restricted to a specific GPU architecture.

For each quantization group g, consisting of 32 weights, LSMXQ computes the quantization error for each candidate bitwidth b as a Hessian-weighted sum of squared errors (SSE):

$$E_{g,b} = \sum_{j\in g} \tilde{s}_{c(j)} \left(W_j - \hat{W}_{j,b}\right)^2,$$

where W_j and $\hat{W}_{j,b}$ denote the original and reconstructed weights, respectively, $c(j)$ denotes the input column containing weight j, and $\tilde{s}_{c(j)}$ is the normalized Hessian-derived sensitivity of that column. The resulting $E_{g,b}$ values are used as the costs in a constrained dynamic-programming allocation, which jointly selects the bitwidth of each group subject to an exact total-bit budget.

Specifically, each squared reconstruction error is weighted by the normalized Hessian-derived sensitivity of its corresponding input column, yielding a cost that approximates the second-order impact of quantization on the reconstruction objective. These costs form a discrete cost matrix C, where

$$C_{g,b} = E_{g,b},$$

and each entry represents the cost of assigning candidate bitwidth b to quantization group g. The resulting cost matrix is then used by the dynamic-programming optimization to determine the optimal bitwidth assignment within each packing block under the specified bit budget and hardware-aware packing constraints.

The bit-allocation problem is then solved using a dynamic-programming knapsack formulation. Algorithm 1 summarizes the allocation procedure.

For high-batch vLLM serving, an offline conversion transforms the original LSMXQ representation into the format required by the optimized NVFP4 execution path. The original representation uses two scale components: a per-16-weight E4M3 scale and a per-256-weight FP16 scale. These are transformed into a new per-16-weight E4M3 scale together with the FP32 scaling parameter required by the NVFP4 GEMM. For projections with a shared layer-level scaling factor, the original scales are decomposed accordingly between the weight-side E4M3 scales and the GEMM-level FP32 scale. For merged projections with component-specific scaling factors, such as QKV and gated-up projections, these factors are incorporated into the per-16-weight E4M3 scales, allowing the entire projection to be executed with a single NVFP4 GEMM invocation. This conversion is specific to the vLLM execution path; batch-1 inference with GPT-Fast[1] uses the native LSMXQ representation directly.

After scale fusion, the resulting per-16-weight E4M3 scale factors are reorganized into the swizzled layout required by the NVFP4 GEMM. This transformation applies only to the scale-factor tensor; the packed E2M1 weight values remain unchanged. The resulting scale tensor therefore matches the hardware-oriented layout expected by the Blackwell low-precision GEMM kernel, allowing the converted representation to be consumed directly without runtime scale reorganization.

[1] https://github.com/meta-pytorch/gpt-fast

Algorithm 1 Hardware-Aware Bitwidth Allocation

Require: Quantization groups $g = 1, \ldots, G$, candidate bitwidths $\mathcal{B}$, group costs $E_{g,b}$, bit costs $q_{g,b}$, target total bit budget Q, packing constraint $\mathcal{P}$
Ensure: Bitwidth assignment $b_1, \ldots, b_G$

```
Initialize DP[0, 0, 0] ← 0 and all other states to +∞
for g = 1, ..., G do
    for each feasible state (q, s) do
        for each b ∈ B do
            Compute the updated budget q' = q + q_{g,b}
            Compute the updated segment state s'
            if (q', s') satisfies the packing constraint P then
                DP[g, q', s'] ← min(DP[g, q', s'], DP[g − 1, q, s] + E_{g,b})
            end if
        end for
    end for
end for
Recover b_1, ..., b_G by backtracking from the optimal feasible state
return b_1, ..., b_G
```

Table 1: Target configurations for hardware-aware bitwidth allocation.

Target Average Bitwidth	Weights	Total Bit Budget	Quantization Groups
2.0	512	1024 bits	16
2.5	2048	5120 bits	64
3.0	1024	3072 bits	32

For batch-1 inference with GPT-Fast, the model is executed directly in its native format using the LSMXQ kernel implementation. For high-batch vLLM serving, the preprocessed scale representation is used by an additional decode step integrated into the execution pipeline, which progressively decodes the weights into the E2M1 representation required by the optimized NVFP4 GEMM kernel. This allows the mixed-bit representation to leverage the established NVFP4 execution path without requiring the model to be loaded in a fully decoded representation.

3.2 Hardware-Aware Representation Considerations

The bitwidth allocation is further constrained by the hardware-friendly organization of the quantized weights in memory. Each quantization group contains 32 weights, and the mixed-bit representation is organized around 1024-bit memory units to enable coalesced access. Accordingly, the dynamic programming formulation uses target-specific problem sizes that align the allocated bit budget with these memory units. Table 1 summarizes the resulting target-specific configurations.

For the 2-bit configuration, the fixed-width representation naturally aligns with the 1024-bit memory boundary. For the 2.5- and 3-bit configurations, different combinations of group bitwidths can satisfy the same target average bitwidth while producing different memory alignments. LSMXQ therefore incorporates the coalescing constraint directly into the dynamic programming formulation. Candidate allocations that cause a 1024-bit boundary to fall within a quantization group are excluded, and among the remaining coalesced allocations, the one with the lowest Hessian-weighted reconstruction error is selected. The resulting groups are then repacked so that the data loaded by each thread corresponds to the 32-bit portion of the weights it processes. This organization preserves quantization-group boundaries across the packing units and allows threads within a warp to operate on their respective weight data without requiring cross-thread reconstruction of groups split across memory boundaries.

3.3 Runtime Execution

LSMXQ uses a precomputed pairwise representation that organizes quantized values in pairs for efficient decoding. This transformation does not change the quantized values or the resulting computation; it only reorganizes their representation to reduce runtime instructions.

During execution, each lane recovers the 2-bit width code of its assigned group from a packed 32-bit metadata word, with 16 groups per word and 2 bits per group selecting one of four bitwidths. The group's bit offset within the packed index stream is then computed using a SWAR prefix-scan rather than an explicit sequential traversal. Specifically, the metadata word is decomposed into its low and high bit-planes using bitmasking, and popcount operations over the two planes, weighted by 1 and 2 respectively, yield the cumulative bit offset. Using this offset and the group's bitwidth, the corresponding bit field is extracted from the packed 64-bit index words and used to index one of four bitwidth-specific regions of a shared-memory lookup table. The lookup directly produces the decoded pairwise representation, with two 4-bit weights packed per byte, which is consumed by the subsequent GEMM.

4 EXPERIMENTS

We evaluate LSMXQ across dense transformer language models ranging from 7B to 72B parameters, covering the Llama (Touvron et al., 2023a;b; Team, 2024) and Qwen (Qwen et al., 2025; Yang et al., 2025) model families. For controlled baseline comparisons, we benchmark LSMXQ against state-of-the-art post-training quantization methods including GPTQ (Frantar et al., 2022), SliM-LLM (Huang et al., 2025), and QTIP (Tseng et al., 2024b) at scales up to 32B parameters using a unified calibration distribution. We adopt QTIP as the representative vector/trellis quantization baseline, as it extends and improves upon earlier codebook-based frameworks such as QuIP (Chee et al., 2023) and QuIP# (Tseng et al., 2024a).

We evaluate both algorithmic compression fidelity and real-world execution efficiency. Specifically, inference benchmarks are conducted across single-token generation (batch size 1) using GPT-Fast and high-throughput batched serving environments in vLLM (Kwon et al., 2023), measuring memory consumption and end-to-end token generation throughput.

4.1 EXPERIMENTAL SETUP

For controlled comparisons with external quantization methods, we evaluate models up to 32B parameters using the same calibration and evaluation setup.

Calibration and Evaluation. Quantization quality is evaluated using perplexity on WikiText-2 (Merity et al., 2017) and zero-shot accuracy on ARC-Challenge, ARC-Easy (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2020). To ensure a strictly controlled and reproducible comparison, all zero-shot evaluations are conducted using the LM-Evaluation-Harness framework (Gao et al., 2023) following the exact evaluation pipeline and task configurations established by SliM-LLM (Huang et al., 2025). LSMXQ is evaluated at 2-, 2.5-, and 3-bit average weight precision.

Hardware and Inference. We conduct kernel-level and end-to-end inference experiments across different execution and serving configurations. These experiments evaluate both batch-one and larger-batch workloads to characterize the performance of the proposed quantized representations.

4.2 CALIBRATION DATASET ANALYSIS

We first investigate the effect of the calibration distribution on quantization quality. In addition to WikiText-2 (Merity et al., 2017), we evaluate a heterogeneous calibration mixture spanning general web text, educational content, conventional language-modeling corpora, and long-form books, using C4 (Dodge et al., 2021), FineWeb-Edu (Penedo et al., 2024), PTB (Marcus et al., 1993), OpenWebText (Gokaslan & Cohen, 2019), and PG19 (Rae et al., 2020).

Across the evaluated models and methods, the mixed calibration generally produces better zero-shot results, particularly at lower bitwidths. Results provided in Table 2 indicate that the calibration distribution can materially affect low-bit quantization quality. Therefore, for the remaining experiments, we use the mixed calibration distribution with the default settings of each method as a common calibration protocol for fair comparison.

Detailed per-model, per-method, and per-task results comparing WikiText-2 and mixed calibration distributions are provided in Appendix A (Tables 30–36).

Table 2: Which calibration set wins on average zero-shot accuracy, `mix` vs. `wikitext2`, per method/model/bit-width.

Bit	Method	LLaMA-7b	LLaMA2-7b	Qwen2.5-7b
3b	GPTQ	mix	mix	mix
3b	SliM-LLM	mix	wt2	mix
3b	QTIP	mix	mix	mix
3b	**LSMXQ**	mix	mix	mix
2.5b	**LSMXQ**	mix	mix	mix
2b	GPTQ	mix	mix	mix
2b	SliM-LLM	mix	mix	mix
2b	QTIP	mix	wt2	mix
2b	**LSMXQ**	mix	mix	wt2

Table 3: LSMXQ zero-shot average accuracy and retention (%) relative to FP16/BF16, across LLaMa 2/3 and Qwen2.5 models.

Model	FP16/BF16	3-bit (%)	2.5-bit (%)	2-bit (%)
LLaMA2-7b	0.7044	0.6833 (97.0)	0.6626 (94.1)	0.6247 (88.7)
LLaMA3-8b	0.7419	0.7309 (98.5)	0.7036 (94.8)	0.6249 (84.2)
Qwen2.5-7b	0.7413	0.7408 (99.9)	0.7156 (96.6)	0.6694 (90.3)
LLaMA2-13b	0.7325	0.7262 (99.1)	0.7176 (98.0)	0.6773 (92.5)
Qwen2.5-14b	0.7726	0.7541 (97.6)	0.7585 (98.2)	0.7210 (93.3)
Qwen2.5-32b	0.7711	0.7792 (101.1)	0.7657 (99.3)	0.7592 (98.5)
LLaMA2-70b	0.7779	0.7733 (99.4)	0.7649 (98.3)	0.7496 (96.4)
LLaMA3-70b	0.8094	0.7700 (95.1)	0.6825 (84.3)	0.5609 (69.3)
Qwen2.5-72b	0.8031	0.7990 (99.5)	0.8066 (100.4)	0.7800 (97.1)

4.3 Quantization Quality

We compare LSMXQ with GPTQ, SliM-LLM, and QTIP under the common calibration protocol described in Section 4.2. Table 4 reports aggregate zero-shot accuracy across models ranging from 7B to 32B parameters, while overall retention rates are summarized in Table 43. As reported in Table 3, LSMXQ retains up to 98.5% of FP16 accuracy at 2-bit precision (Qwen2.5-32B) and matches baseline performance at 2.5- and 3-bit regimes (up to 100.4% on Qwen2.5-72B and 101.1% on Qwen2.5-32B). This retention trend generally strengthens with model scale, with LLaMA3-70B as a notable exception Table 41. Detailed per-model breakdown tables across all evaluated model families, including scaling results up to 70B parameters, are provided in Appendix A.

At 2-bit precision, LSMXQ substantially outperforms GPTQ and SliM-LLM and remains competitive with QTIP. The advantage becomes more pronounced as model size increases, with LSMXQ achieving the highest average zero-shot accuracy in the evaluated 30B and 32B configurations. At 3 bits, the differences between methods are smaller, but LSMXQ remains competitive and achieves the highest average zero-shot accuracy in most evaluated configurations. Furthermore, on large-scale benchmarks such as LLaMA-2-70B (Table 29 in Appendix A), LSMXQ consistently matches or exceeds the accuracy of vector-quantized baselines.

The 2.5-bit configuration provides an intermediate operating point between the more aggressive 2-bit setting and 3-bit precision. It consistently improves over the corresponding 2-bit LSMXQ results while requiring less storage than 3-bit representations, providing an effective accuracy–compression trade-off.

LSMXQ achieves competitive perplexity across the evaluated settings, attaining the lowest values in most 3-bit configurations. In the 2-bit regime, QTIP achieves lower perplexity on most models, while LSMXQ remains competitive or superior in average zero-shot accuracy.

Table 4: Average zero-shot accuracy of LSMXQ, GPTQ, SliM-LLM, and QTIP across 7B–32B models. Only LSMXQ is evaluated at 2.5-bit.

Bit	Method	LLaMA-7b	LLaMA2-7b	LLaMA3-8b	Qwen2.5-7b	LLaMA-13b	LLaMA2-13b	Qwen2.5-14b	LLaMA-30b	Qwen2.5-32b
3b	GPTQ	0.6579	0.6472	0.6531	0.6955	0.7063	0.7057	0.7099	0.7357	0.7628
3b	SliM-LLM	0.5925	0.6104	0.6922	0.7114	0.6971	0.7128	0.7580	0.7430	0.7527
3b	QTIP	0.6343	0.6328	0.7211	0.7368	0.6543	0.6545	0.7754	0.6860	0.7675
3b	**LSMXQ**	**0.6778**	**0.6833**	**0.7309**	**0.7408**	**0.7118**	**0.7262**	**0.7541**	**0.7501**	**0.7792**
2.5b	**LSMXQ**	**0.6535**	**0.6626**	**0.7036**	**0.7156**	**0.6967**	**0.7176**	**0.7585**	**0.7308**	**0.7657**
2b	GPTQ	0.3870	0.3828	0.3711	0.3956	0.4177	0.4277	0.4034	0.4465	0.4456
2b	SliM-LLM	0.4844	0.4851	0.4082	0.5315	0.5711	0.5866	0.5689	0.5994	0.6625
2b	QTIP	0.6092	0.6058	0.6196	0.6788	0.6123	0.6234	0.7150	0.6662	0.7518
2b	**LSMXQ**	**0.6057**	**0.6247**	**0.6249**	**0.6694**	**0.6434**	**0.6773**	**0.7210**	**0.6939**	**0.7592**

Table 5: Single-sequence (M=1) decode throughput (tok/s), RTX 5090: FP16 vs. LSMXQ 2-bit.

Model	FP16	LSMXQ 2-bit	Speedup
LLaMA-7b	117.4	341.6	**2.91**×
LLaMA-13b	62.2	216.0	**3.47**×
LLaMA-30b	OOM	101.5	—
LLaMA-65b	OOM	51.2	—
Qwen3-8b	102.4	255.0	**2.49**×
Qwen3-14b	56.6	173.4	**3.06**×

4.4 KERNEL-LEVEL PERFORMANCE

We evaluate LSMXQ inference performance using GPT-Fast and vLLM across modern GPU architectures. As summarized in Table 5 (RTX 5090) and Table 6 (RTX PRO 6000), at batch size one, LSMXQ 2-bit achieves up to 3.71× speedup over FP16, with the relative throughput gain consistently scaling with model size. Furthermore, the compact memory footprint substantially reduces resident VRAM overhead (e.g., enabling LLaMA-30B to run within 9.93 GiB at 2-bit), allowing larger parameter models to fit comfortably onto a single workstation GPU where standard FP16 encounters out-of-memory (OOM) bottlenecks.

Table 6: Single-sequence (M=1) decode throughput (tok/s), GPU-RTXPRO6000: FP16 vs. LSMXQ across bit-widths, with resident VRAM footprint.

Model	FP16	2-bit	2.5-bit	3-bit
LLaMA-7b	106.1	**268.6** (2.00 GiB)	**245.8** (2.39 GiB)	**230.2** (2.81 GiB)
LLaMA-13b	56.1	**175.5** (3.92 GiB)	**145.0** (4.68 GiB)	**140.1** (5.51 GiB)
LLaMA-30b	23.0	**85.4** (9.93 GiB)	**71.7** (11.84 GiB)	**67.5** (13.94 GiB)
Qwen3-8b	92.1	**196.0** (2.14 GiB)	**188.4** (2.56 GiB)	**176.4** (3.01 GiB)
Qwen3-14b	51.8	**145.0** (4.06 GiB)	**127.2** (4.87 GiB)	**119.2** (5.73 GiB)
Qwen3-32b	23.0	**76.4** (9.58 GiB)	**67.4** (11.50 GiB)	**62.5** (13.54 GiB)

In production LLM serving, a central benefit of ultra-low-bit weight quantization is reducing static model residency to liberate GPU VRAM directly for runtime states, most critically, the dynamic KV cache. To assess this operational headroom, we evaluate end-to-end serving throughput under varying KV cache allocations in vLLM. Here, B denotes the scheduler's maximum concurrent batch size, NP represents the total number of submitted prompts, and the KV cache budget reflects the GPU memory dedicated to key–value states.

As reported in Tables 7 and 8, at matched KV cache allocations, LSMXQ delivers 1.89×–2.10× the throughput of unquantized BF16 (e.g., 11,369.40 vs. 5,413.57 tok/s on Qwen3-8B at 8 GiB KV) while retaining up to 87.5% of the raw compute throughput of native, hardware-accelerated NVFP4. Crucially, despite a modest throughput delta at identical KV budgets, LSMXQ's substantially reduced weight footprint unlocks a significantly larger feasible serving envelope before out-of-memory (OOM) limits are reached.

On Qwen3-8B, native NVFP4 triggers an OOM fault at 22 GiB of KV cache, whereas LSMXQ maintains robust execution at 14,711.07 tok/s. This capacity advantage becomes even more decisive

at larger scales: on Qwen3-32B, native NVFP4 exhausts memory at merely 6 GiB of KV cache, while LSMXQ scales smoothly up to 10 GiB, supporting 2.5× the maximum feasible KV allocation of NVFP4 and achieving 1.48× its peak observed throughput (2,888.89 vs. 1,952.65 tok/s). These results show that, at matched KV cache budgets, LSMXQ incurs a moderate throughput reduction relative to native NVFP4, while its smaller weight footprint substantially extends the feasible KV cache allocation and peak serving throughput under memory-constrained settings.

Table 7: Throughput comparison across KV cache sizes on Qwen3-8B ($B = 256$, $NP = 512$) GPU-RTX 5090.

KV (GiB)	BF16	NVFP4	LSMXQ(2b)
4	4382.71	9899.90	8303.14
8	5413.57	13232.66	11369.40
16	OOM	16839.16	14684.31
20	OOM	16791.15	14690.46
21	OOM	16821.11	14698.59
22	OOM	**OOM**	14711.07

Table 8: Throughput comparison across KV cache sizes on Qwen3-32B ($B = 256$, $NP = 1024$) GPU-RTX 5090.

KV (GiB)	NVFP4	LSMXQ(2b)
4	1952.65	1208.96
6	OOM	2114.55
10	OOM	2888.89

5 Conclusion

We presented LSMXQ, a hardware-aware post-training quantization framework for ultra-low-bit LLM inference. LSMXQ uses Hessian-weighted reconstruction error to allocate 1-, 2-, 3-, and 4-bit representations across weight groups, jointly considering quantization sensitivity, bitwidth budgets, and hardware-aware memory constraints. The resulting representation is organized for coalesced memory access and NVFP4-compatible execution, enabling the compressed weights to be decoded efficiently without requiring a conventional FP16 weight representation.

Across dense LLMs ranging from 7B to 72B parameters, LSMXQ maintains strong quantization quality at 2-, 2.5-, and 3-bit average precision while substantially reducing model memory requirements. The resulting representation also translates into practical inference gains, achieving significant speedups over FP16 at batch-one decoding and approaching native NVFP4 performance in larger-batch serving. Overall, LSMXQ demonstrates that ultra-low-bit quantization can jointly optimize numerical accuracy, bitwidth allocation, memory layout, and runtime execution rather than treating quantization as an isolated compression problem. This hardware-aware formulation provides a foundation for further optimization of low-bit decoding kernels and efficient deployment of compressed LLMs across local and server-scale inference systems.

AI USE STATEMENT

In this work, we used generative AI tools for drafting parts of the paper, identifying and analyzing relevant literature, sourcing/searching for information, creating or editing software code and providing feedback on research methodology or experiments.

We did not use generative AI tools for conceiving the original research idea, devising the experimentation methodology, conceiving the novel quantization approach and hardware aware optimization algorithms.

We have reviewed all AI-assisted work. Furthermore, all references were collected manually from approved online resources.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We believe that our paper submission does not raise questions regarding the Code of Ethics. Specifically:

- Contribute to society and to human well-being: We comply.
- Uphold high standards of scientific excellence: We comply.
- Avoid harm: Non-applicable
- Be honest, trustworthy and transparent: We comply
- Be fair and take action to avoid discrimination: Non-applicable
- Respect the work required to produce new ideas and artefacts: We comply
- Respect privacy: Non-applicable
- Honour confidentiality: Non-applicable

REPRODUCIBILITY STATEMENT

To ensure complete reproducibility, the core implementation, quantization scripts, and benchmark configurations are provided in an anonymized repository in the supplementary material and will be made publicly available upon acceptance.

REFERENCES

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. Quip: 2-bit quantization of large language models with guarantees. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/0df38cd13520747e1e64e5b123a78ef8-Abstract-Conference.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,*

NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pp. 2924–2936. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1300. URL https://doi.org/10.18653/v1/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. *CoRR*, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1286–1305, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.98. URL https://aclanthology.org/2021.emnlp-main.98/.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/1caf09c9f4e6b0150b06a07e77f2710c-Abstract-Conference.html.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: accurate post-training quantization for generative pre-trained transformers. *CoRR*, abs/2210.17323, 2022. doi: 10.48550/ARXIV.2210.17323. URL https://doi.org/10.48550/arXiv.2210.17323.

Elias Frantar, Roberto L. Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. MARLIN: mixed-precision auto-regressive parallel inference on large language models. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, PPoPP 2025, Las Vegas, NV, USA, March 1-5, 2025*, pp. 239–251. ACM, 2025. doi: 10.1145/3710848.3710871. URL https://doi.org/10.1145/3710848.3710871.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Oorgan, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://doi.org/10.5281/zenodo.10256836.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qinshuo Liu, Xianglong Liu, Luca Benini, Michele Magno, Shiming Zhang, and Xiaojuan Qi. Slim-llm: Salience-driven mixed-precision quantization for large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/huang25aa.html.

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 Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann, and Jonathan Mace (eds.), *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pp. 611–626. ACM, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

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 Phillip B. Gibbons, Gennady Pekhimenko, and Christopher De Sa (eds.), *Proceedings of the Seventh Annual Conference on Machine Learning and Systems, MLSys 2024, Santa Clara, CA, USA, May 13-16, 2024*. mlsys.org, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330, 1993.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Byj72udxe.

NVIDIA. Pretraining large language models with NVFP4. *CoRR*, abs/2509.25149, 2025. doi: 10.48550/ARXIV.2509.25149. URL https://doi.org/10.48550/arXiv.2509.25149.

Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin A. Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html.

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.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Dusan Stosic, Venmugil Elango, Maximilian Golub, Alexander Heinecke, Phil James-Roxby, Dharmesh Jani, Gaurav Kolhe, Martin Langhammer, Ada Li, Levi Melnick, Maral Mesmakhosroshahi, Andres Rodriguez, Michael Schulte, Rasoul Shafipour, Lei Shao, Michael Y. Siu, Pradeep Dubey, Paulius Micikevicius, Maxim Naumov, Colin Verilli, Ralph Wittig, Doug Burger, and Eric S. Chung. Microscaling data formats for deep learning. *CoRR*, abs/2310.10537, 2023. doi: 10.48550/ARXIV.2310.10537. URL https://doi.org/10.48550/arXiv.2310.10537.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8732–8740. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6399. URL https://doi.org/10.1609/aaai.v34i05.6399.

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023a. doi: 10.48550/ARXIV.2302.13971. URL https://doi.org/10.48550/arXiv.2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b. doi: 10.48550/ARXIV.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48630–48656. PMLR / OpenReview.net, 2024a. URL https://proceedings.mlr.press/v235/tseng24a.html.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De Sa. QTIP: quantization with trellises and incoherence processing. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024b. URL http://papers.nips.cc/paper_files/paper/2024/hash/6de2e84b8da47bb2eb5e2ac96c63d2b0-Abstract-Conference.html.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472. URL https://doi.org/10.18653/v1/p19-1472.

A APPENDIX

Table 9: LLaMA-7b: LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	—	0.4471	0.7294	0.7505	0.7621	0.7911	0.7001	**0.6967**
GPTQ	3b	0.3916	0.6591	0.7355	0.7211	0.7612	0.6788	**0.6579**
SliM-LLM	3b	0.3746	0.4769	0.6657	0.6676	0.7437	0.6267	**0.5925**
QTIP	3b	0.4027	0.5084	0.7581	0.7041	0.7737	0.6590	**0.6343**
LSMXQ	**3b**	**0.4249**	**0.7125**	**0.7306**	**0.7280**	**0.7840**	**0.6867**	**0.6778**
LSMXQ	**2.5b**	**0.3763**	**0.6755**	**0.7257**	**0.7022**	**0.7715**	**0.6701**	**0.6535**
GPTQ	2b	0.2389	0.2719	0.4875	0.3086	0.5092	0.5059	**0.3870**
SliM-LLM	2b	0.2756	0.3973	0.6226	0.4281	0.6257	0.5572	**0.4844**
QTIP	2b	0.3703	0.4844	0.7382	0.6669	0.7563	0.6393	**0.6092**
LSMXQ	**2b**	**0.3319**	**0.6338**	**0.7055**	**0.6023**	**0.7307**	**0.6298**	**0.6057**

Table 10: LLaMA-7b: WikiText2 PPL, LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	PPL
FP16	—	5.677
GPTQ	3b	6.7243
SliM-LLM	3b	6.6572
QTIP	3b	6.0445
LSMXQ	**3b**	**6.075**
LSMXQ	**2.5b**	**6.690**
GPTQ	2b	80.16
QTIP	2b	6.7307
LSMXQ	**2b**	**8.773**

Table 11: LLaMA2-7b: LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	—	0.4625	0.7458	0.7765	0.7599	0.7911	0.6906	**0.7044**
GPTQ	3b	0.3840	0.6271	0.7303	0.7198	0.7661	0.6559	**0.6472**
SliM-LLM	3b	0.3805	0.5143	0.6804	0.6919	0.7557	0.6393	**0.6104**
QTIP	3b	0.3916	0.5219	0.7422	0.7060	0.7573	0.6780	**0.6328**
LSMXQ	**3b**	**0.4249**	**0.7290**	**0.7550**	**0.7372**	**0.7802**	**0.6732**	**0.6833**
LSMXQ	**2.5b**	**0.4027**	**0.6793**	**0.7263**	**0.7067**	**0.7709**	**0.6898**	**0.6626**
GPTQ	2b	0.2747	0.2681	0.4700	0.2756	0.4995	0.5091	**0.3828**
SliM-LLM	2b	0.2713	0.3876	0.6211	0.4408	0.6469	0.5430	**0.4851**
QTIP	2b	0.3652	0.4907	0.6979	0.6740	0.7486	0.6582	**0.6058**
LSMXQ	**2b**	**0.3592**	**0.6364**	**0.7275**	**0.6293**	**0.7416**	**0.6543**	**0.6247**

Table 12: LLaMA2-7b: WikiText2 PPL, LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	PPL
FP16	—	5.47
GPTQ	3b	6.5990
SliM-LLM	3b	6.3490
QTIP	3b	5.8402
LSMXQ	**3b**	**5.805**
LSMXQ	**2.5b**	**6.397**
GPTQ	2b	100.89
QTIP	2b	6.5467
LSMXQ	**2b**	**8.085**

Table 13: LLaMA3-8b: LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	—	0.5350	0.7769	0.8138	0.7917	0.8079	0.7261	**0.7419**
GPTQ	3b	0.4070	0.6561	0.7263	0.7229	0.7519	0.6543	**0.6531**
SliM-LLM	3b	0.4522	0.7155	0.7813	0.7426	0.7563	0.7056	**0.6922**
QTIP	3b	0.4812	0.7681	0.7966	0.7619	0.7927	0.7261	**0.7211**
LSMXQ	**3b**	**0.5068**	**0.7723**	**0.8156**	**0.7622**	**0.7960**	**0.7324**	**0.7309**
LSMXQ	**2.5b**	**0.4795**	**0.7428**	**0.7783**	**0.7293**	**0.7829**	**0.7088**	**0.7036**
GPTQ	2b	0.2628	0.2622	0.4300	0.2566	0.5201	0.4949	**0.3711**
SliM-LLM	2b	0.2278	0.2976	0.5336	0.3219	0.5468	0.5217	**0.4082**
QTIP	2b	0.3720	0.6141	0.6893	0.6431	0.7339	0.6654	**0.6196**
LSMXQ	**2b**	**0.3677**	**0.6301**	**0.7205**	**0.6372**	**0.7421**	**0.6519**	**0.6249**

Table 14: LLaMA3-8b: WikiText2 PPL, LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	PPL
FP16	—	6.14
GPTQ	3b	10.7088
SliM-LLM	3b	8.1333
QTIP	3b	7.1291
LSMXQ	**3b**	**7.027**
LSMXQ	**2.5b**	**8.127**
GPTQ	2b	1214.58
QTIP	2b	11.6301
LSMXQ	**2b**	**11.892**

Table 15: Qwen2.5-7b: LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	—	0.5102	0.7740	0.8471	0.7895	0.7976	0.7293	**0.7413**
GPTQ	3b	0.4642	0.7256	0.7859	0.7474	0.7791	0.6709	**0.6955**
SliM-LLM	3b	0.4923	0.7235	0.8370	0.7494	0.7835	0.6827	**0.7114**
QTIP	3b	0.5128	0.7875	0.8517	0.7637	0.7954	0.7096	**0.7368**
LSMXQ	**3b**	**0.5205**	**0.7955**	**0.8428**	**0.7699**	**0.7965**	**0.7198**	**0.7408**
LSMXQ	**2.5b**	**0.4881**	**0.7563**	**0.8266**	**0.7423**	**0.7769**	**0.7032**	**0.7156**
GPTQ	2b	0.2423	0.2908	0.4291	0.3434	0.5490	0.5193	**0.3956**
SliM-LLM	2b	0.3148	0.4975	0.6028	0.5327	0.6866	0.5549	**0.5315**
QTIP	2b	0.4778	0.7424	0.7343	0.6628	0.7557	0.7001	**0.6788**
LSMXQ	**2b**	**0.4403**	**0.7323**	**0.7462**	**0.6716**	**0.7677**	**0.6582**	**0.6694**

Table 16: Qwen2.5-7b: WikiText2 PPL, LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	PPL
FP16	—	6.80
GPTQ	3b	8.3132
SliM-LLM	3b	7.7988
QTIP	3b	7.3704
LSMXQ	**3b**	**7.291**
LSMXQ	**2.5b**	**7.910**
GPTQ	2b	74.22
QTIP	2b	11.2762
LSMXQ	**2b**	**9.681**

Table 17: Qwen3-8b: LSMXQ vs. SliM-LLM (GPTQ and QTIP are not evaluated on Qwen3-8B, as their public implementations do not support the Qwen3 architecture).

Method	Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	—	0.5648	0.8093	0.8664	0.7493	0.7769	0.6780	**0.7408**
SliM-LLM	3b	0.4872	0.7235	0.8505	0.7111	0.7595	0.6803	**0.7020**
LSMXQ	**3b**	**0.5256**	**0.7605**	**0.8612**	**0.7225**	**0.7688**	**0.6819**	**0.7201**
LSMXQ	**2.5b**	**0.5171**	**0.7731**	**0.8474**	**0.6917**	**0.7617**	**0.6851**	**0.7127**
SliM-LLM	2b	0.2995	0.4533	0.6208	0.4622	0.6502	0.5454	**0.5052**
LSMXQ	**2b**	**0.4189**	**0.6646**	**0.8217**	**0.6098**	**0.7285**	**0.6235**	**0.6445**

Table 18: Qwen3-8b: WikiText2 PPL, LSMXQ vs. SliM-LLM (GPTQ and QTIP are not evaluated on Qwen3-8B, as their public implementations do not support the Qwen3 architecture).

Method	Bit	PPL
FP16	—	9.728
SliM-LLM	3b	11.3756
LSMXQ	**3b**	**10.585**
LSMXQ	**2.5b**	**11.364**
SliM-LLM	2b	32.62
LSMXQ	**2b**	**13.443**

Table 19: LLaMA-13b: LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	—	0.4778	0.7483	0.7792	0.7905	0.8014	0.7277	**0.7208**
GPTQ	3b	0.4582	0.7357	0.7651	0.7662	0.7927	0.7198	**0.7063**
SliM-LLM	3b	0.4514	0.7319	0.7538	0.7607	0.7873	0.6977	**0.6971**
QTIP	3b	0.4232	0.5804	0.6930	0.7451	0.7802	0.7040	**0.6543**
LSMXQ	**3b**	**0.4616**	**0.7357**	**0.7697**	**0.7727**	**0.7954**	**0.7356**	**0.7118**
LSMXQ	**2.5b**	**0.4403**	**0.7109**	**0.7651**	**0.7520**	**0.7911**	**0.7206**	**0.6967**
GPTQ	2b	0.2509	0.3279	0.4976	0.3653	0.5707	0.4941	**0.4177**
SliM-LLM	2b	0.3029	0.5480	0.6832	0.5541	0.7133	0.6251	**0.5711**
QTIP	2b	0.3976	0.5354	0.6376	0.6751	0.7612	0.6669	**0.6123**
LSMXQ	**2b**	**0.3899**	**0.6675**	**0.6590**	**0.6883**	**0.7699**	**0.6859**	**0.6434**

Table 20: LLaMA-13b: WikiText2 PPL, LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	PPL
FP16	—	5.091
GPTQ	3b	5.69
SliM-LLM	3b	5.54
QTIP	3b	5.3676
LSMXQ	**3b**	**5.361**
LSMXQ	**2.5b**	**5.751**
GPTQ	2b	23.54
SliM-LLM	2b	10.62
QTIP	2b	6.7242
LSMXQ	**2b**	**6.922**

Table 21: LLaMA2-13b: LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	—	0.4923	0.7748	0.8058	0.7939	0.8052	0.7230	**0.7325**
GPTQ	3b	0.4428	0.7487	0.7856	0.7679	0.7797	0.7096	**0.7057**
SliM-LLM	3b	0.4701	0.7348	0.7924	0.7664	0.7987	0.7143	**0.7128**
QTIP	3b	0.4283	0.5715	0.7122	0.7475	0.7878	0.6796	**0.6545**
LSMXQ	**3b**	**0.4923**	**0.7614**	**0.8009**	**0.7764**	**0.8003**	**0.7261**	**0.7262**
LSMXQ	**2.5b**	**0.4778**	**0.7504**	**0.8021**	**0.7600**	**0.7889**	**0.7261**	**0.7176**
GPTQ	2b	0.2637	0.3089	0.5437	0.3642	0.5539	0.5320	**0.4277**
SliM-LLM	2b	0.3396	0.5758	0.6798	0.5832	0.7149	0.6267	**0.5866**
QTIP	2b	0.3916	0.5396	0.7000	0.6748	0.7579	0.6764	**0.6234**
LSMXQ	**2b**	**0.4198**	**0.7037**	**0.7713**	**0.6967**	**0.7709**	**0.7017**	**0.6773**

Table 22: LLaMA2-13b: WikiText2 PPL, LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	PPL
FP16	—	4.88
GPTQ	3b	5.51
SliM-LLM	3b	5.30
QTIP	3b	5.1670
LSMXQ	**3b**	**5.125**
LSMXQ	**2.5b**	**5.486**
GPTQ	2b	307.53
SliM-LLM	2b	9.93
QTIP	2b	6.5311
LSMXQ	**2b**	**6.760**

Table 23: Qwen2.5-14b: LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	—	0.5896	0.7912	0.8526	0.8292	0.8210	0.7522	**0.7726**
GPTQ	3b	0.5444	0.7984	0.5963	0.7862	0.8090	0.7253	**0.7099**
SliM-LLM	3b	0.5683	0.8068	0.8465	0.7959	0.8058	0.7245	**0.7580**
QTIP	3b	0.5802	0.8279	0.8590	0.8047	0.8166	0.7640	**0.7754**
LSMXQ	**3b**	**0.5222**	**0.7601**	**0.8529**	**0.8150**	**0.8025**	**0.7719**	**0.7541**
LSMXQ	**2.5b**	**0.5452**	**0.8211**	**0.8156**	**0.7932**	**0.8074**	**0.7687**	**0.7585**
GPTQ	2b	0.2338	0.3199	0.4979	0.3294	0.5305	0.5091	**0.4034**
SliM-LLM	2b	0.3302	0.5143	0.6535	0.5885	0.6991	0.6275	**0.5689**
QTIP	2b	0.4983	0.7744	0.7847	0.7311	0.7704	0.7309	**0.7150**
LSMXQ	**2b**	**0.4932**	**0.7605**	**0.8309**	**0.7404**	**0.7775**	**0.7238**	**0.7210**

Table 24: Qwen2.5-14b: WikiText2 PPL, LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	PPL
FP16	—	5.25
GPTQ	3b	7.01
SliM-LLM	3b	6.63
QTIP	3b	6.1898
LSMXQ	**3b**	**6.061**
LSMXQ	**2.5b**	**6.683**
GPTQ	2b	55.73
SliM-LLM	2b	14.81
QTIP	2b	8.2918
LSMXQ	**2b**	**7.972**

Table 25: LLaMA-30b: LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	—	0.5290	0.7896	0.8278	0.8265	0.8226	0.7585	**0.7590**
GPTQ	3b	0.5026	0.7761	0.7954	0.7953	0.7943	0.7506	**0.7357**
SliM-LLM	3b	0.5026	0.7643	0.8272	0.8083	0.8041	0.7514	**0.7430**
QTIP	3b	0.4573	0.5939	0.7379	0.7838	0.7982	0.7451	**0.6860**
LSMXQ	**3b**	**0.5205**	**0.7904**	**0.8156**	**0.8120**	**0.8118**	**0.7506**	**0.7501**
LSMXQ	**2.5b**	**0.5043**	**0.7563**	**0.8031**	**0.7870**	**0.7943**	**0.7395**	**0.7308**
GPTQ	2b	0.2423	0.3716	0.5361	0.4177	0.5827	0.5288	**0.4465**
SliM-LLM	2b	0.3439	0.6098	0.6514	0.6177	0.7242	0.6496	**0.5994**
QTIP	2b	0.4266	0.5614	0.7505	0.7470	0.7840	0.7277	**0.6662**
LSMXQ	**2b**	**0.4582**	**0.7483**	**0.7183**	**0.7312**	**0.7786**	**0.7277**	**0.6939**

Table 26: LLaMA-30b: WikiText2 PPL, LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	PPL
FP16	—	4.10
GPTQ	3b	4.9150
QTIP	3b	4.3226
LSMXQ	**3b**	**4.394**
LSMXQ	**2.5b**	**4.829**
QTIP	2b	5.1246
LSMXQ	**2b**	**5.807**

Table 27: Qwen2.5-32b: LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	—	0.5563	0.7832	0.8713	0.8408	0.8226	0.7522	**0.7711**
GPTQ	3b	0.5725	0.7971	0.8606	0.8137	0.8194	0.7135	**0.7628**
SliM-LLM	3b	0.5324	0.7719	0.8425	0.8222	0.8161	0.7309	**0.7527**
QTIP	3b	0.5444	0.7769	0.8798	0.8150	0.8194	0.7695	**0.7675**
LSMXQ	**3b**	**0.5811**	**0.8030**	**0.8862**	**0.8312**	**0.8264**	**0.7474**	**0.7792**
LSMXQ	**2.5b**	**0.5819**	**0.7946**	**0.8141**	**0.8183**	**0.8183**	**0.7672**	**0.7657**
GPTQ	2b	0.2884	0.3754	0.5177	0.4268	0.5827	0.4822	**0.4456**
SliM-LLM	2b	0.4326	0.6557	0.7676	0.6984	0.7541	0.6669	**0.6625**
QTIP	2b	0.5230	0.7740	0.8752	0.7757	0.8069	0.7561	**0.7518**
LSMXQ	**2b**	**0.5563**	**0.8001**	**0.8590**	**0.7861**	**0.8025**	**0.7514**	**0.7592**

Table 28: Qwen2.5-32b: WikiText2 PPL, LSMXQ vs. GPTQ, SliM-LLM, and QTIP.

Method	Bit	PPL
FP16	—	4.97
QTIP	3b	5.8983
LSMXQ	**3b**	**5.500**
LSMXQ	**2.5b**	**5.962**
QTIP	2b	7.1963
LSMXQ	**2b**	**6.867**

Table 29: Full zero-shot result table for LSMXQ vs. SOTA (QTIP, QuIP#) on LLaMA2-70b, across calibration sets and bit-widths.

Model	Calibration	Bit	ArcC	ArcE	BoolQ	HellaSwag	PIQA	WinoGrande	Avg
LSMXQ	mix	2b	0.5324	0.7879	0.8086	0.7861	0.7982	0.7845	**0.7496**
LSMXQ	mix	2.5b	0.5444	0.7900	0.8401	0.8176	0.8215	0.7758	**0.7649**
LSMXQ	mix	3b	0.5648	0.8022	0.8324	0.8318	0.8286	0.7798	**0.7733**
LSMXQ	wikitext2	2b	0.4872	0.7673	0.8135	0.7770	0.7911	0.7664	**0.7337**
LSMXQ	wikitext2	2.5b	0.5444	0.7824	0.8410	0.8162	0.8156	0.7743	**0.7623**
LSMXQ	wikitext2	3b	0.5666	0.8064	0.8272	0.8288	0.8210	0.7822	**0.7720**
QTIP (HuggingFace)	wikitext2	2b	0.4642	0.5842	0.7642	0.7941	0.8041	0.7498	**0.6934**
QTIP (HuggingFace)	wikitext2	3b	0.4770	0.5934	0.7758	0.8041	0.8112	0.7703	**0.7053**
QuIP# (HuggingFace)	wikitext2	2b	0.4659	0.5791	0.7823	0.7954	0.7960	0.7459	**0.6941**
QuIP# (HuggingFace)	wikitext2	3b	0.4838	0.5951	0.7486	0.8050	0.8030	0.7640	**0.6999**

Table 30: LSMXQ, LLaMA2-7b: `mix` vs. `wikitext2` calibration.

Bit	Calib	PPL	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
2b	mix	8.085	0.3592	0.6364	0.7275	0.6293	0.7416	0.6543	**0.6247**
2b	wt2	7.6465	0.3490	0.6128	0.6957	0.6121	0.7111	0.6496	**0.6050**
2.5b	mix	6.397	0.4027	0.6793	0.7263	0.7067	0.7709	0.6898	**0.6626**
2.5b	wt2	6.7505	0.3737	0.6700	0.6972	0.6709	0.7557	0.6551	**0.6371**
3b	mix	5.805	0.4249	0.7290	0.7550	0.7372	0.7802	0.6732	**0.6833**
3b	wt2	6.4273	0.4121	0.6949	0.7018	0.7047	0.7633	0.6748	**0.6586**

Table 31: LSMXQ, Qwen2.5-7b: `mix` vs. `wikitext2` calibration.

Bit	Calib	PPL	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
2b	mix	9.681	0.4403	0.7323	0.7462	0.6716	0.7677	0.6582	**0.6694**
2b	wt2	9.1531	0.4411	0.7125	0.8086	0.6541	0.7524	0.6559	**0.6708**
2.5b	mix	7.910	0.4881	0.7563	0.8266	0.7423	0.7769	0.7032	**0.7156**
2.5b	wt2	8.3693	0.4556	0.7462	0.8110	0.7082	0.7655	0.6993	**0.6976**
3b	mix	7.291	0.5205	0.7955	0.8428	0.7699	0.7965	0.7198	**0.7408**
3b	wt2	7.7794	0.4650	0.6957	0.8104	0.7422	0.7758	0.6969	**0.6977**

Table 32: LSMXQ, LLaMA-7b (LLaMA-1): `mix` vs. `wikitext2` calibration.

Bit	Calib	PPL	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
2b	mix	8.7726	0.3319	0.6338	0.7055	0.6023	0.7307	0.6298	**0.6057**
2b	wt2	9.2965	0.2782	0.4832	0.6208	0.5080	0.6785	0.6006	**0.5282**
2.5b	mix	6.6895	0.3763	0.6755	0.7257	0.7022	0.7715	0.6701	**0.6535**
2.5b	wt2	8.7882	0.3106	0.5114	0.5309	0.5885	0.7127	0.6235	**0.5463**
3b	mix	6.0750	0.4249	0.7125	0.7306	0.7280	0.7840	0.6867	**0.6778**
3b	wt2	7.8073	0.3439	0.5614	0.6985	0.6308	0.7334	0.6385	**0.6011**

Table 33: External baseline (GPTQ): `mix` vs. `wikitext2` calibration.

Model	Bit	Calib	PPL	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
LLaMA-7b	2b	mix	80.16	0.2389	0.2719	0.4875	0.3086	0.5092	0.5059	**0.3870**
LLaMA-7b	2b	wt2	80.74	0.2474	0.2883	0.4150	0.2901	0.5218	0.5107	**0.3789**
LLaMA-7b	3b	mix	6.7243	0.3916	0.6591	0.7355	0.7211	0.7612	0.6788	**0.6579**
LLaMA-7b	3b	wt2	6.7875	0.4113	0.6574	0.6948	0.7225	0.7644	0.6867	**0.6562**
LLaMA2-7b	2b	mix	100.89	0.2747	0.2681	0.4700	0.2756	0.4995	0.5091	**0.3828**
LLaMA2-7b	2b	wt2	105.13	0.2799	0.2702	0.4382	0.2626	0.4782	0.5036	**0.3721**
LLaMA2-7b	3b	mix	6.5990	0.3840	0.6271	0.7303	0.7198	0.7661	0.6559	**0.6472**
LLaMA2-7b	3b	wt2	6.4450	0.3754	0.6402	0.7300	0.7140	0.7524	0.6630	**0.6458**
LLaMA3-8b	2b	mix	1214.58	0.2628	0.2622	0.4300	0.2566	0.5201	0.4949	**0.3711**
LLaMA3-8b	2b	wt2	459.68	0.2585	0.2584	0.3969	0.2613	0.5082	0.5020	**0.3642**
LLaMA3-8b	3b	mix	10.7088	0.4070	0.6561	0.7263	0.7229	0.7519	0.6543	**0.6531**
LLaMA3-8b	3b	wt2	10.4154	0.4113	0.6427	0.5865	0.7021	0.7416	0.6614	**0.6243**
Qwen2.5-7b	2b	mix	74.22	0.2423	0.2908	0.4291	0.3434	0.5490	0.5193	**0.3956**
Qwen2.5-7b	2b	wt2	41.49	0.2568	0.2795	0.4196	0.2959	0.5120	0.4957	**0.3766**
Qwen2.5-7b	3b	mix	8.3132	0.4642	0.7256	0.7859	0.7474	0.7791	0.6709	**0.6955**
Qwen2.5-7b	3b	wt2	8.0472	0.4138	0.6595	0.7972	0.7384	0.7758	0.6582	**0.6738**

Table 34: External baseline (SliM-LLM): zero-shot accuracy, `mix` vs. `wikitext2` calibration.

Model	Bit	Calib	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
LLaMA-7b	2b	mix	0.2756	0.3973	0.6226	0.4281	0.6257	0.5572	**0.4844**
LLaMA-7b	2b	wt2	0.2423	0.3316	0.5890	0.3344	0.5707	0.5438	**0.4353**
LLaMA-7b	3b	mix	0.3746	0.4769	0.6657	0.6676	0.7437	0.6267	**0.5925**
LLaMA-7b	3b	wt2	0.3635	0.4777	0.6474	0.6255	0.7280	0.6267	**0.5781**
LLaMA2-7b	2b	mix	0.2713	0.3876	0.6211	0.4408	0.6469	0.5430	**0.4851**
LLaMA2-7b	2b	wt2	0.2730	0.3779	0.5954	0.4003	0.6230	0.5549	**0.4708**
LLaMA2-7b	3b	mix	0.3805	0.5143	0.6804	0.6919	0.7557	0.6393	**0.6104**
LLaMA2-7b	3b	wt2	0.3916	0.5269	0.6859	0.6978	0.7590	0.6575	**0.6198**
LLaMA3-8b	2b	mix	0.2278	0.2976	0.5336	0.3219	0.5468	0.5217	**0.4082**
LLaMA3-8b	2b	wt2	0.2167	0.3262	0.4003	0.2874	0.5419	0.5162	**0.3814**
LLaMA3-8b	3b	mix	0.4522	0.7155	0.7813	0.7426	0.7563	0.7056	**0.6922**
LLaMA3-8b	3b	wt2	0.4232	0.6911	0.7526	0.7297	0.7693	0.7024	**0.6781**
Qwen2.5-7b	2b	mix	0.3148	0.4975	0.6028	0.5327	0.6866	0.5549	**0.5315**
Qwen2.5-7b	2b	wt2	0.2875	0.4794	0.5113	0.4662	0.6436	0.5564	**0.4908**
Qwen2.5-7b	3b	mix	0.4923	0.7235	0.8370	0.7494	0.7835	0.6827	**0.7114**
Qwen2.5-7b	3b	wt2	0.5043	0.7332	0.7768	0.7518	0.7775	0.6732	**0.7028**
Qwen3-8b	2b	mix	0.2995	0.4533	0.6208	0.4622	0.6502	0.5454	**0.5052**
Qwen3-8b	2b	wt2	0.2594	0.3649	0.6000	0.4234	0.6023	0.5485	**0.4664**
Qwen3-8b	3b	mix	0.4872	0.7235	0.8505	0.7111	0.7595	0.6803	**0.7020**
Qwen3-8b	3b	wt2	0.4710	0.7205	0.8569	0.7029	0.7595	0.6582	**0.6949**

Table 35: External baseline (SliM-LLM): WikiText2 PPL, `mix` vs. `wikitext2` calibration.

Model	Bit	Calib	PPL
LLaMA-7b	3b	mix	6.6572
LLaMA-7b	3b	wt2	7.4900
LLaMA2-7b	3b	mix	6.3490
LLaMA2-7b	3b	wt2	6.6505
LLaMA3-8b	2b	wt2	69.3616
LLaMA3-8b	3b	mix	8.1333
LLaMA3-8b	3b	wt2	7.6337
Qwen2.5-7b	3b	mix	7.7988
Qwen2.5-7b	3b	wt2	7.6919
Qwen3-8b	2b	mix	32.6247
Qwen3-8b	2b	wt2	29.8617
Qwen3-8b	3b	mix	11.3756
Qwen3-8b	3b	wt2	11.4555

Table 36: External baseline (QTIP): `mix` vs. `wikitext2` calibration.

Model	Bit	Calib	PPL	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
LLaMA-7b	2b	mix	6.7307	0.3703	0.4844	0.7382	0.6669	0.7563	0.6393	**0.6092**
LLaMA-7b	2b	wt2	6.4518	0.3729	0.4971	0.7324	0.6615	0.7448	0.6346	**0.6072**
LLaMA-7b	3b	mix	6.0445	0.4027	0.5084	0.7581	0.7041	0.7737	0.6590	**0.6343**
LLaMA-7b	3b	wt2	5.9616	0.3831	0.4861	0.7064	0.7047	0.7748	0.6685	**0.6206**
LLaMA2-7b	2b	mix	6.5467	0.3652	0.4907	0.6979	0.6740	0.7486	0.6582	**0.6058**
LLaMA2-7b	2b	wt2	6.3278	0.3660	0.5097	0.7101	0.6645	0.7448	0.6630	**0.6097**
LLaMA2-7b	3b	mix	5.8402	0.3916	0.5219	0.7422	0.7060	0.7573	0.6780	**0.6328**
LLaMA2-7b	3b	wt2	5.7730	0.3882	0.5324	0.6872	0.7082	0.7639	0.6669	**0.6245**
LLaMA3-8b	2b	mix	11.6301	0.3720	0.6141	0.6893	0.6431	0.7339	0.6654	**0.6196**
LLaMA3-8b	2b	wt2	10.4077	0.3797	0.6073	0.6844	0.6268	0.7323	0.6756	**0.6177**
LLaMA3-8b	3b	mix	7.1291	0.4812	0.7681	0.7966	0.7619	0.7927	0.7261	**0.7211**
LLaMA3-8b	3b	wt2	6.9688	0.5094	0.7820	0.8003	0.7627	0.7954	0.7348	**0.7308**
Qwen2.5-7b	2b	mix	11.2762	0.4778	0.7424	0.7343	0.6628	0.7557	0.7001	**0.6788**
Qwen2.5-7b	2b	wt2	14.5622	0.4130	0.6439	0.8024	0.6600	0.7459	0.6803	**0.6576**
Qwen2.5-7b	3b	mix	7.3704	0.5128	0.7875	0.8517	0.7637	0.7954	0.7096	**0.7368**
Qwen2.5-7b	3b	wt2	7.3447	0.5034	0.7807	0.8431	0.7640	0.7960	0.7080	**0.7325**

Table 37: LSMXQ zero-shot accuracy, LLaMA-65b (LLaMA-1).

Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	0.5546	0.7980	0.8480	0.8412	0.8232	0.7743	**0.7732**
3b	0.5512	0.7887	0.8483	0.8300	0.8205	0.7719	**0.7684**
2.5b	0.5341	0.7883	0.8456	0.8162	0.8172	0.7766	**0.7630**
2b	0.4974	0.7647	0.8180	0.7768	0.8041	0.7640	**0.7375**

Table 38: LSMXQ zero-shot accuracy, Qwen3-14b.

Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	0.6024	0.8287	0.8933	0.7884	0.7982	0.7277	**0.7731**
3b	0.6007	0.8123	0.8933	0.7704	0.7982	0.7253	**0.7667**
2.5b	0.5503	0.7849	0.8881	0.7491	0.7922	0.7230	**0.7479**
2b	0.5137	0.7567	0.8407	0.6974	0.7655	0.7024	**0.7127**

Table 39: LSMXQ zero-shot accuracy, Qwen3-32b.

Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
BF16	0.6092	0.8308	0.8633	0.8255	0.8199	0.7332	**0.7803**
3b	0.6118	0.8338	0.8804	0.8131	0.8069	0.7245	**0.7784**
2.5b	0.5930	0.7963	0.8801	0.8027	0.8009	0.7096	**0.7638**
2b	0.5870	0.8089	0.8758	0.7585	0.7824	0.6977	**0.7517**

Table 40: LSMXQ zero-shot accuracy, LLaMA2-70b.

Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	0.5734	0.8106	0.8376	0.8384	0.8275	0.7798	**0.7779**
3b	0.5648	0.8022	0.8324	0.8318	0.8286	0.7798	**0.7733**
2.5b	0.5444	0.7900	0.8401	0.8176	0.8215	0.7758	**0.7649**
2b	0.5324	0.7879	0.8086	0.7861	0.7982	0.7845	**0.7496**

Table 41: LSMXQ zero-shot accuracy, LLaMA3-70b. At 2-bit, ArcC/ArcE/PIQA collapse toward chance level while BoolQ/HS/Wino remain robust; a similar collapse on LLaMA3-70B has been reported by (Tseng et al., 2024b).

Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
FP16	0.6433	0.8586	0.8523	0.8491	0.8455	0.8074	**0.8094**
3b	0.5333	0.7845	0.8480	0.8424	0.7965	0.8153	**0.7700**
2.5b	0.3601	0.5816	0.8465	0.8215	0.6904	0.7948	**0.6825**
2b	0.2577	0.2896	0.8141	0.7360	0.5310	0.7372	**0.5609**

Table 42: LSMXQ zero-shot accuracy, Qwen2.5-72b.

Bit	ArcC	ArcE	BoolQ	HS	PIQA	Wino	**Avg**
BF16	0.6271	0.8316	0.8908	0.8609	0.8346	0.7735	**0.8031**
3b	0.6126	0.8207	0.8930	0.8490	0.8292	0.7893	**0.7990**
2.5b	0.6254	0.8489	0.8954	0.8414	0.8335	0.7948	**0.8066**
2b	0.5734	0.8081	0.8850	0.8127	0.8188	0.7822	**0.7800**

Table 43: LSMXQ zero-shot average accuracy and retention (%) relative to FP16/BF16, across all evaluated models.

Model	FP16/BF16	3-bit (%)	2.5-bit (%)	2-bit (%)
LLaMA-7b	0.6967	0.6778 (97.3)	0.6535 (93.8)	0.6057 (86.9)
LLaMA-13b	0.7208	0.7118 (98.8)	0.6967 (96.7)	0.6434 (89.3)
LLaMA-30b	0.7590	0.7501 (98.8)	0.7308 (96.3)	0.6939 (91.4)
LLaMA-65b	0.7732	0.7684 (99.4)	0.7630 (98.7)	0.7375 (95.4)
Qwen3-8b	0.7408	0.7201 (97.2)	0.7127 (96.2)	0.6445 (87.0)
Qwen3-14b	0.7731	0.7667 (99.2)	0.7479 (96.7)	0.7127 (92.2)
Qwen3-32b	0.7803	0.7784 (99.8)	0.7638 (97.9)	0.7517 (96.3)
LLaMA2-7b	0.7044	0.6833 (97.0)	0.6626 (94.1)	0.6247 (88.7)
LLaMA3-8b	0.7419	0.7309 (98.5)	0.7036 (94.8)	0.6249 (84.2)
Qwen2.5-7b	0.7413	0.7408 (99.9)	0.7156 (96.6)	0.6694 (90.3)
LLaMA2-13b	0.7325	0.7262 (99.1)	0.7176 (98.0)	0.6773 (92.5)
Qwen2.5-14b	0.7726	0.7541 (97.6)	0.7585 (98.2)	0.7210 (93.3)
Qwen2.5-32b	0.7711	0.7792 (101.1)	0.7657 (99.3)	0.7592 (98.5)
LLaMA2-70b	0.7779	0.7733 (99.4)	0.7649 (98.3)	0.7496 (96.4)
LLaMA3-70b	0.8094	0.7700 (95.1)	0.6825 (84.3)	0.5609 (69.3)
Qwen2.5-72b	0.8031	0.7990 (99.5)	0.8066 (100.4)	0.7800 (97.1)