BATQuant: Outlier-Resilient MXFP4 Quantization via Learnable Block-wise Optimization

Ji-Fu Li[1], Manyi Zhang[1*], Xiaobo Xia[2], Han Bao[1], Haoli Bai[1], Zhenhua Dong[1], and Xianzhi Yu[1]

[1] Huawei Technologies
[2] University of Science and Technology of China
{lijifu4, zhangmanyi6}@huawei.com

Abstract. Microscaling floating-point (MXFP) formats have emerged as a promising standard for deploying Multi-modal Large Language Models (MLLMs) and Large Language Models (LLMs) on modern accelerator architectures. However, existing Post-Training Quantization (PTQ) methods, particularly rotation-based techniques designed for integer formats, suffer from severe performance collapse when applied to MXFP4. Recent studies attribute this failure to a fundamental format mismatch: global orthogonal rotations inadvertently transfer outlier energy across quantization blocks, inducing new outliers that disrupt local block-wise scaling, while often creating bimodal activation distributions that underutilize the limited quantization range. To address these issues, we propose BATQuant (**B**lock-wise **A**ffine **T**ransformation), which restricts transformations to align with MXFP granularity to prevent cross-block outlier propagation, while relaxing orthogonality constraints to optimize distribution shaping. To ensure parameter efficiency, we introduce **G**lobal and **P**rivate **K**ronecker (GPK) decomposition to effectively reduces storage and runtime overhead and incorporate Block-wise Learnable Clipping to suppress residual outliers. Extensive experiments on both MLLMs and LLMs demonstrate that BATQuant establishes new state-of-the-art results under aggressive W4A4KV16 configurations, recovering up to **96.43%** of full-precision performance on multimodal benchmarks and clearly outperforming existing methods across diverse tasks.

Keywords: Multi-modal Large Language Models · Large Language Models · Quantization · MXFP

1 Introduction

Multi-modal Large Language Models (MLLMs) and Large Language Models (LLMs) have recently revolutionized artificial intelligence, demonstrating remarkable capabilities in bridging visual perception with linguistic reasoning [4,

* Corresponding author.

14, 26, 33, 46, 50, 54, 62, 65]. From autonomous driving to medical image analysis, these models are increasingly deployed in real-world scenarios where low latency and memory efficiency are paramount [27, 30, 51, 58, 67]. However, the ever-growing scale of MLLMs and LLMs, often comprising billions of parameters, imposes prohibitive costs on memory bandwidth and computational resources, hindering their deployment on edge devices and resource-constrained platforms.

Post-Training Quantization (PTQ) has emerged as a key solution to mitigate these costs [29]. While integer quantization has been widely studied, the recent emergence of microscaling floating-point formats (MXFP) offers a promising alternative [2, 41]. Supported by next-generation hardware [1, 7, 47], MXFP4 utilizes block-wise scaling to better accommodate the long-tailed distributions inherent in activations, theoretically offering superior dynamic range compared to fixed-point formats. Despite this hardware readiness, achieving accurate 4-bit quantization for MLLMs under the MXFP format remains an unsolved challenge [64, 66].

While existing state-of-the-art PTQ methods are predominantly designed for INT formats [21, 22, 34, 39, 43, 49, 52], their applicability to MXFP formats is contested. Specifically, popular rotation-based techniques (e.g., QuaRot [3] and SpinQuant [32]), which excel in INT4 by spreading outliers via orthogonal transformations, suffer from severe performance collapse when applied to MXFP4 [11, 36]. Recent studies [11, 44] have attributed this failure to the incompatibility between global rotations and the fine-grained quantization settings of MXFP, and further propose block-wise rotation transformation methods. However, these approaches still fail to mitigate extreme outliers within certain blocks, and the Hadamard transform further introduces a bimodal distribution problem (see Figure 2a).

To bridge this gap, in this paper, we introduce BATQuant. The core of our method is the *Block-wise Affine Transformation* (BAT). Unlike global rotations, BAT restricts the transformation scope to align strictly with the MXFP quantization granularity (e.g., 32 elements). This design prevents the cross-block energy transfer of outliers, ensuring that each block's scaling factor accurately captures its local dynamic range. Moreover, we relax the orthogonality constraint and learn the optimal affine matrices tailored to the MXFP format to minimize quantization error. To address the storage overhead caused by learnable block-wise affine transformations, we further introduce the *Global and Private Kronecker* (GPK) decomposition that drastically reduces parameter counts by sharing a global transformation basis across blocks while retaining block-specific private components. Finally, we incorporate *Block-wise Learnable Clipping*, which dynamically adapts thresholds to suppress residual outliers within quantization blocks.

We validate our BATQuant extensively on both MLLMs and LLMs. Our method achieves **near-lossless** performance on `W4A8KV16` with an accuracy recovery rate exceeding **99%**. Furthermore, it establishes the new state-of-the-art results under aggressive `W4A4KV16` configurations, recovering up to **96.43%** on

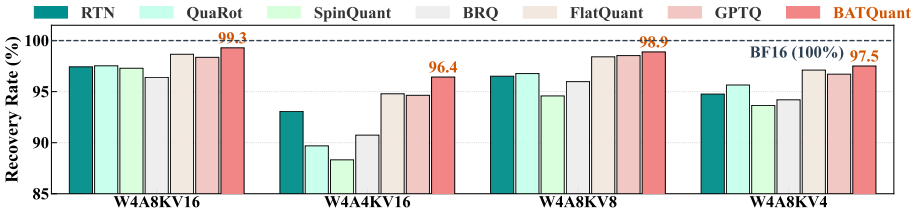


Fig. 1: Quantization performance on Qwen3-VL-8B-Instruct across various methods. Our method yields superior results compared to baselines across all bit-width settings. The advantage is particularly substantial in the W4A4 setting, where our method clearly outperforms existing methods.

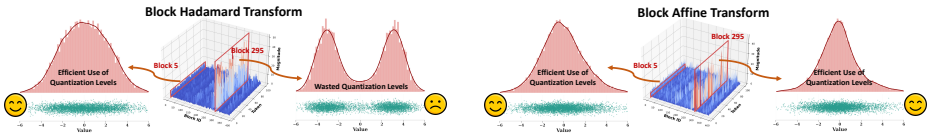
multimodal benchmarks, significantly outperforming existing methods (see Figure 1). Our main contributions are summarized as follows:

- We propose BATQuant, featuring a Block-wise Affine Transformation that aligns with MXFP granularity to prevent energy transfer across blocks and address the bimodal distribution problem for effective quantization. Additionally, we incorporate Global and Private Kronecker decomposition for parameter efficiency.
- We evaluate BATQuant on both MLLMs and LLMs, such as Qwen3-8B-VL-Instruct [4] and Qwen3-8B [59], covering a wide range of challenging settings. The effectiveness is validated, ranging from knowledge understanding to complex reasoning benchmarks, setting new state-of-the-art results in most scenarios.

2 Preliminary

Microscaling Floating-Point Definition. The MXFP, proposed by OCP [41], is a family of floating-point formats that employ block-wise quantization. An MXFP format is defined by three components: a sign bit (S), an exponent (E), and a mantissa (M). Each MXFP format uses a fixed block size of 32 elements, with all values in a block sharing a common scaling factor represented in UE8M0 format (8-bit exponent, no mantissa). The standard MXFP4 (E2M1) format uses 1 sign bit, 2 exponent bits, and 1 mantissa bit. This configuration represents 7 distinct positive values: $\{0.5, 1.0, 1.5, 2.0, 3.0, 4.0, 6.0\}$, along with their negatives and zero. MXFP8 offers two variants, E4M3 and E5M2. Here, we adopt E4M3 for MXFP8, as a larger mantissa width is more crucial for the performance of fine-grained quantization [6, 37].

Related Work. Initial research on LLM quantization primarily explored integer-based formats [12, 16, 24, 25, 56, 60]. As NVFP and MXFP formats gain hardware support, quantization accuracy under these formats is also drawing increasing attention [15, 19, 28, 57, 63]. Prior work has shown that MXFP8 achieves lossless



(a) BRQ **(b)** BATQuant

Fig. 2: Activation distributions for the `down_proj` module in layer 35 of Qwen3-8B. The central 3D plots illustrate the activations after transformation. We specifically extract Block 5 (without outliers) and Block 295 (with extreme outliers), and visualize the values after scaling factor division but prior to rounding. (a) After applying the block Hadamard transform, block 295 exhibits a bimodal distribution, leading to inefficient utilization of the bit width. (b) After the block affine transformation, block 295 shows reduced magnitude compared to subplot (a) while effectively leveraging the floating-point quantization grids.

quantization, whereas MXFP4 suffers from significant accuracy degradation [64]. For the low-bit scenarios, e.g., 4-bit quantization, outliers are considered as a severe impediment. The primary methods for suppressing outliers include rotation transformations [17,22,48] and affine transformations [34,45]. The rotation-based methods, such as QuaRot [3] and SpinQuant [32], unlike their success in INT4 quantization, underperform even basic RTN when applied to MXFP4. Such global rotations mix dimensional information, suppressing outliers and kurtosis, thus disrupting the local statistical properties of fine-grained formats. To address the incompatibility between rotation-based techniques and MXFP4, BRQ [44] is proposed to utilize block-wise rotation quantization to mitigate outliers and prevent amplifying small-value blocks. MR-GPTQ [11], a GPTQ variant optimized for FP4, similarly employs block-wise Hadamard transforms and format-specific adjustments to accommodate FP4's unique properties. Affine transformation-based methods, such as FlatQuant [45], overcome the energy-conservation constraints inherent in rotation-based transformations and enhance quantization accuracy by employing affine transformations. Nevertheless, previous methods still suffer from significant accuracy degradation on MXFP4 quantization, particularly on complex reasoning tasks [64].

Observations and Motivation. We find that block-wise rotation still struggles to suppress extreme outliers in specific blocks, and the Hadamard transform further introduces a bimodal distribution problem. Specifically, we visualize the activations after block-wise Hadamard transformation on Qwen3-8B in Figure 2a. We observe that although the block-wise Hadamard transform reduces the magnitude for the vast majority of blocks, since Hadamard matrices are composed of $\{+1, -1\}$ values, certain blocks with extreme outliers exhibit a bimodal distribution. This results in wasted bit-width and introduces larger quantization errors [10]. Therefore, to address these challenges, we propose BATQuant. As shown in Figure 2b, BATQuant effectively alleviates outliers, while ensuring that the post-transformation data distribution remains amenable to floating-point quantization.

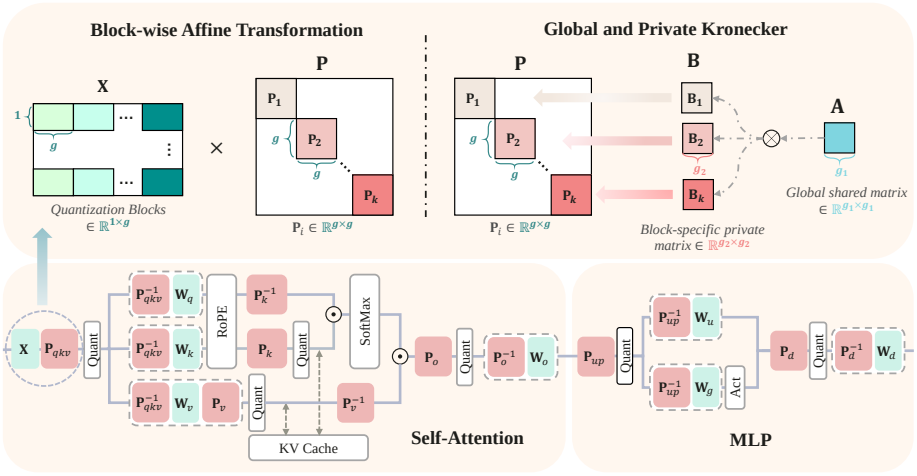


Fig. 3: The overall framework of BATQuant. *Bottom:* Integration of BATQuant into the Transformer architecture. Weight-side transformations are fused offline into the linear layers, while activation-side transformations are applied online. *Top:* Exemplary view of the *Block-wise Affine Transformation*, where inputs are partitioned into MXFP-aligned blocks. Each block transformation is decomposed via the *Global and Private Kronecker*.

3 Method

In this section, we present BATQuant with the framework illustrated in Figure 3. We first introduce learning optimal block-wise affine transformations in Section 3.1. Afterward, we discuss its integration with the Transformer architecture in Section 3.2. Note that we provide a detailed algorithm flow of our BATQuant in Appendix B.

3.1 Block-wise Affine Transformation

Consider a standard linear layer computation $\mathbf{Y} = \mathbf{X}\mathbf{W}^\top$, where $\mathbf{X} \in \mathbb{R}^{S\times N}$ represents activations and $\mathbf{W} \in \mathbb{R}^{M\times N}$ denotes weights. The primary objective is to find the best affine transformation $\mathbf{P}^\star \in \mathbb{R}^{N\times N}$ for each linear layer to quantize:

$$\mathbf{P}^\star = \arg\min_{\mathbf{P}} ||\mathbf{Y} - \mathcal{Q}(\mathbf{X}\mathbf{P})\mathcal{Q}(\mathbf{P}^{-1}\mathbf{W}^\top)||_F^2.$$

Instead of learning a single global matrix, we partition the transformation matrix into k disjoint blocks aligned with the MXFP quantization granularity g (e.g., $g = 32$). We then construct a block-diagonal affine matrix:

$$\mathbf{P} = \mathrm{diag}(\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_k), \quad \text{where } \mathbf{P}_i \in \mathbb{R}^{g\times g}, N = k \cdot g. \tag{1}$$

Table 1: Comparison of decomposition methods on parameter counts and computational cost. For the example parameter count, we set the hidden dim $N = 4096$ and the MXFP quantization granularity $g = 32$. The size of decomposed matrix $\mathbf{A}_i$ and $\mathbf{B}_i$ are set to $g_1 = 8$ and $g_2 = 4$. The reported *MatMul Complexity* refers to the computational cost of the activation transformation $\mathbf{XP}$.

Method	Decomposition	MatMul Complexity	# Params of $\mathbf{P}$	Example Count
FlatQuant	Kronecker	$\mathcal{O}(SN^{\frac{3}{2}})$	$2N$	8,192
Ours	w/o	$\mathcal{O}(SNg)$	$N \cdot g$	131,072
	Naive Kronecker	$\mathcal{O}(SN(g_1 + g_2))$	$k \cdot (g_1^2 + g_2^2)$	10,240
	GPK	$\mathcal{O}(SN(g_1 + g_2))$	$g_1^2 + k \cdot g_2^2$	2,112

Here, each $\mathbf{P}_i$ is an independent and learnable affine transformation applied solely within the i-th quantization block. By restricting the transformation scope to the size of the MXFP block, our method ensures that outlier redistribution occurs only locally. This preserves the statistical independence of each quantization block, allowing the MXFP scaling factors to accurately capture the dynamic range of each block without interference from outliers of other blocks.

Global and Private Kronecker. Although the block-diagonal structure of $\mathbf{P}$ introduces inherent sparsity, the total number of learnable parameters remains $N \cdot g$. For large-scale models, storing such a matrix for every layer still incurs a significant memory cost. A straightforward approach to mitigate this is to apply Kronecker product decomposition to each $\mathbf{P}_i$, factorizing it into two smaller matrices $\mathbf{B}_i \otimes \mathbf{A}_i$, where $\mathbf{A}_i \in \mathbb{R}^{g_1 \times g_1}, \mathbf{B}_i \in \mathbb{R}^{g_2 \times g_2}$. The g_1 and g_2 respectively denote the size of $\mathbf{A}_i$ and $\mathbf{B}_i$ and we have MXFP quantization granularity $g = g_1 \cdot g_2$. We refer to this as *Naive Kronecker*. However, since the block size g is typically small (e.g., 32 in MXFP formats), the reduction in parameter count is marginal.

To address this limitation, we propose *Global and Private Kronecker* (GPK). GPK decomposes each $\mathbf{P}_i$ into the product of a *global shared matrix* $\mathbf{A}$ and a *block-specific private matrix* $\mathbf{B}_i$:

$$\mathbf{P}_i = \mathbf{B}_i \otimes \mathbf{A}, \quad \forall i \in \{1, \ldots, k\}, \tag{2}$$

where $\mathbf{A}$ is shared across all k blocks and $\mathbf{B}_i$ is unique to the i-th block. This design drastically reduces the storage requirement from $k \cdot (g_1^2 + g_2^2)$ to $g_1^2 + k \cdot g_2^2$. As shown in Table 1, GPK significantly reduces the storage overhead, reducing the parameter count by more than 74% and 79% compared to FlatQuant and Naive Kronecker. Additionally, by leveraging the vectorization trick of the Kronecker product, i.e., $\text{vec}(\mathbf{V})(\mathbf{B}_i \otimes \mathbf{A}) = \text{vec}(\mathbf{B}_i^\top \mathbf{V} \mathbf{A})$ for some $\mathbf{V} \in \mathbb{R}^{g_2 \times g_1}$, GPK maintains efficient inference by preserving the low matrix multiplication complexity. Here, we provide the PyTorch-style pseudo code of the forward pass with GPK in Appendix B.

Block-wise Learnable Clipping. While the block-wise affine transformation effectively smooths activation distributions, residual outliers may still persist within the quantization blocks, potentially dominating the quantization range of MXFP formats. To mitigate this, we introduce *Block-wise Learnable Clipping*, a fine-grained strategy that adapts clipping thresholds to the local statistics of each quantization block. For the i-th block, the clipped values $\hat{\mathbf{x}}_i$ (and similarly for weights $\hat{\mathbf{w}}_i$) are computed as:

$$\hat{\mathbf{x}}_i = \text{clip}\left(\mathbf{x}_i,\ \beta_i^{\min},\ \beta_i^{\max}\right), \tag{3}$$

where the dynamic bounds $\beta_i^{\min}$ and $\beta_i^{\max}$ are:

$$\beta_i^{\min} = \sigma(\alpha_i^{\min}) \cdot \min(\mathbf{x}_i), \quad \beta_i^{\max} = \sigma(\alpha_i^{\max}) \cdot \max(\mathbf{x}_i). \tag{4}$$

Here, $\min(\mathbf{x}_i)$ and $\max(\mathbf{x}_i)$ denote the minimum and maximum values within the i-th block, respectively, and $\sigma(\cdot)$ is the sigmoid function constraining the clipping ratios to $(0, 1)$. α_i is the learnable parameter specific to block i.

The Training Objective. Following previous work [45], we optimize the block-wise affine transformations and clipping factors by minimizing the layer-wise quantization errors between the full-precision and quantized outputs over a small calibration set $\mathcal{D}_{\text{cal}}$:

$$\Theta_l^* = \arg\min_{\Theta_l} \mathbb{E}_{\mathbf{X}\sim\mathcal{D}_{\text{cal}}} \left[\left\| \mathcal{F}_l(\mathbf{X}) - \hat{\mathcal{F}}_l(\mathbf{X}; \Theta_l) \right\|_2^2 \right] \tag{5}$$

where $\mathcal{F}_l(\cdot)$ and $\hat{\mathcal{F}}_l(\cdot)$ denote the full-precision layer l and quantized layer l, respectively. Θ_l is abbreviated for all learnable parameters within the quantization block.

3.2 Integration with the Transformer Architecture

We integrate BATQuant into both LLM (Qwen3) and MLLM (Qwen3-VL) architectures by inserting block-wise affine transformations into the transformer block, where the weight-side transformations are merged into the linear layers offline, while the activation-side transformations are applied online during inference. Following the conventional practices, we employ low-bit matrix multiplications for all linear layers, while keeping layer normalization layers, pre-quantization transformations, RoPE embeddings and attention scores in BF16.

MLP Module. In LLM and the text model of MLLM, the MLP module employs two transformation sets, $\mathbf{P}_{up}$ and $\mathbf{P}_{down}$. $\mathbf{P}_{up}$ flattens the activation distribution after LayerNorm before the `up_proj` and `gate_proj` layers. $\mathbf{P}_{down}$ smooths the input to the `down_proj` layer. In the ViT model of MLLM, the MLP module also employs two transformation sets: $\mathbf{P}_{fc1}$ and $\mathbf{P}_{fc2}$. $\mathbf{P}_{fc1}$ flattens the activation distribution after LayerNorm before the `linear_fc1` layers. $\mathbf{P}_{fc2}$ smooths the input to the `linear_fc2` layer. All matrices utilize the GPK decomposition to minimize storage.

Self-Attention Module. In LLM and the text model of MLLM, the Self-Attention module employs four transformations: $\mathbf{P}_{qkv}$, $\mathbf{P}_o$, $\mathbf{P}_k$ and $\mathbf{P}_v$. $\mathbf{P}_{qkv}$ and $\mathbf{P}_o$ flatten the activation distribution before the `qkv_proj` layer and `o_proj` layer respectively. $\mathbf{P}_k$ and $\mathbf{P}_v$ are used to transform the key and value cache head by head, respectively. In the ViT model of MLLM, only $\mathbf{P}_{qkv}$ and $\mathbf{P}_o$ are employed. This is because ViT does not require an autoregressive KV cache mechanism. Consequently, there is no need to store, transform and quantize the key and value states across generation steps.

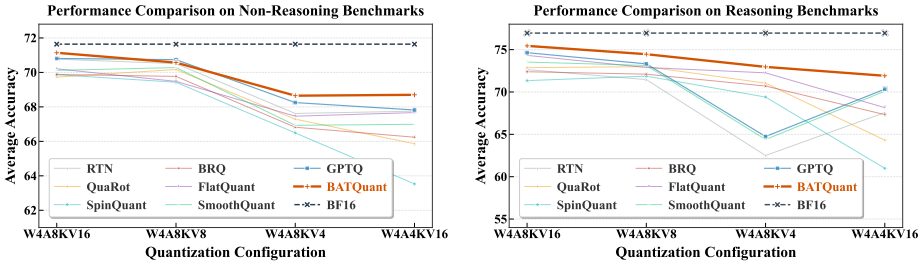
4 Experiments

4.1 Settings

Evaluation and Baselines. We evaluate BATQuant on Qwen3-VL-8B-Instruct (MLLM) [4] and Qwen3-8B (LLM) [59]. We assess quantized models on the following benchmarks: (1) Multimodal benchmarks, including MME [13], OCR-Bench [31], DocVQA [35], RealWorldQA [55], and VLMBlind. (2) Non-reasoning tasks, including PIQA [5], Winogrande [42], Hellaswag [61], ARC-Easy [8], and ARC-Challenge [8]. (3) Reasoning benchmarks, including GSM8K [9], MATH-500 [23], AIME24, AIME25 and GPQA-D [40]. We compare BATQuant against popular post-training quantization methods, including QuaRot [3], SpinQuant [32], BRQ [44], FlatQuant [45], SmoothQuant [56] and GPTQ [12]. More details about benchmarks and baseline methods are provided in Appendix A.

Implementation Details. We implement BATQuant based on Huggingface [53], PyTorch [38]. We adopt the AdamW optimizer with an initial learning rate of 2e-3 and employ a cosine annealing learning rate decay schedule. BATQuant is trained for 5 epochs, and the batch size is set to 4. For GPK, we set the size of the global shared matrix g_1 and block-specific private matrix g_2 to 8 and 4, respectively. For LLM, we use the BF16 model to self-generate data on the Numina-Math-1.5 [20] dataset, and randomly sample 128 text sequences of length 2048 to construct the calibration set. For MLLM, we randomly sample 128 image-text pairs from the GQA [18] dataset to construct the calibration set. Further details about implementation are provided in Appendix A.

Quantization Settings. We evaluate the proposed method on several MXFP-based quantization configurations, including weight-activation quantization and KV cache quantization. For clarity, we denote each configuration using the format W{bits}A{bits}KV{bits}. For example, W4A8KV8 indicates quantizing weights to 4-bit, activations to 8-bit, and KV cache to 8-bit. We empirically observe that combining different methods with GPTQ universally enhances performance. Consequently, unless otherwise specified, the reported results refer to the GPTQ-integrated variants of each method. Detailed comparisons between GPTQ and RTN weight quantizer are provided in Appendix C.



(a) Performance of Qwen3-8B on Non-Reasoning tasks under different quantization settings.

(b) Performance of Qwen3-8B on Reasoning tasks under different quantization settings.

Fig. 4: Performance comparison of different methods on Qwen3-8B across LLM benchmarks under various quantization configurations. The results are categorized into Non-Reasoning (left) and Reasoning (right) tasks.

4.2 Main Results

Here, we present a comprehensive empirical evaluation of BATQuant. Our experiments are designed to answer the following critical questions: (1) Can BATQuant maintain satisfactory performance under aggressive MXFP-based quantization configurations where existing methods fail? (2) How does our approach generalize across modalities (MLLMs vs. LLMs) and task domains, specifically spanning *multimodal understanding* (including document understanding, STEM puzzles, and general VQA) in MLLMs and *linguistic task* (covering non-reasoning and reasoning tasks) in LLMs?

Results on Multimodal Benchmarks. Table 2 summarizes the performance of different post-training quantization methods on the Qwen3-VL-8B-Instruct model across five multimodal benchmarks. As shown in the table, BATQuant consistently establishes state-of-the-art results across all bit-width configurations. Notably, in the aggressive `W4A4KV16` regime, BATQuant achieves an average recovery rate of 96.43%, significantly outperforming the strongest baseline FlatQuant by a margin of 1.64%. Under `W4A8KV16` scenario, BATQuant achieves an average recovery rate of 99.29%, which is the only approach exhibiting a performance degradation of under 1%. This superiority extends to KV cache quantization as well. Under `W4A8KV8` and `W4A8KV4`, our method maintains superior performance with recovery rates of 98.89% and 97.51%, respectively. Such a consistent performance gain is also widely observed across different types of benchmarks, including document understanding, STEM puzzles, and general VQA. We attribute this success to our method's unique capability to mitigate inter-block energy transfer, thereby effectively capturing diverse outlier patterns that conventional methods fail to address.

Results on LLM Benchmarks. To comprehensively evaluate the generalization capability of BATQuant beyond multimodal, we conduct extensive exper-

Table 2: Performance comparison of various quantization methods on multimodal benchmarks across different bit-width configurations (e.g., W4A8KV16, W4A4KV16, W4A8KV8 and W4A8KV4).The recovery rate relative to the BF16 baseline is also provided and the best result in each case is marked in bold.

Bits	Method	MME	OCRBench	DocVQA	RealWorldQA	VLMBlind	*Recovery(%)*
BF16	–	2377	906	95.81	70.98	73.98	100.00
W4A8KV16	RTN	2294	883	94.72	69.80	70.99	97.43
	QuaRot	2327	870	95.07	69.80	71.12	97.53
	SpinQuant	2321	872	94.79	**70.46**	69.82	97.29
	BRQ	2329	865	94.72	70.19	67.18	96.40
	FlatQuant	2351	886	95.31	69.02	**73.90**	98.66
	SmoothQuant	2349	885	94.81	70.06	69.46	97.61
	GPTQ	2346	891	95.03	69.15	72.62	98.36
	BATQuant	**2386**	**893**	**95.55**	70.20	73.14	**99.29**
W4A4KV16	RTN	2243	838	92.70	65.23	66.47	93.07
	QuaRot	2189	810	93.47	64.97	57.62	89.69
	SpinQuant	1994	801	91.79	65.36	60.23	88.32
	BRQ	2147	805	92.94	66.14	62.14	90.74
	FlatQuant	2231	**873**	94.10	65.62	68.86	94.79
	SmoothQuant	2264	862	93.93	**68.89**	66.26	95.01
	GPTQ	2286	849	93.98	66.93	67.29	94.64
	BATQuant	**2360**	864	**94.31**	67.32	**69.70**	**96.43**
W4A8KV8	RTN	2208	878	94.64	69.54	71.01	96.51
	QuaRot	2296	868	95.11	69.02	70.26	96.77
	SpinQuant	2217	832	94.41	68.10	69.04	94.58
	BRQ	2283	867	94.63	69.80	67.36	95.98
	FlatQuant	2353	888	95.12	69.14	72.77	98.41
	SmoothQuant	2317	884	94.72	70.19	68.91	97.19
	GPTQ	2340	885	95.14	**71.11**	71.79	98.53
	BATQuant	**2368**	**890**	**95.47**	69.93	**72.82**	**98.89**
W4A8KV4	RTN	2220	856	94.05	68.50	67.50	94.76
	QuaRot	2280	857	94.66	68.52	68.36	95.65
	SpinQuant	2248	829	94.18	68.63	64.50	93.65
	BRQ	2236	841	94.07	68.63	66.03	94.20
	FlatQuant	2293	884	94.88	**68.76**	70.75	97.11
	SmoothQuant	2283	871	94.39	67.02	66.99	95.13
	GPTQ	2328	867	94.15	68.10	70.81	96.71
	BATQuant	**2332**	**885**	**95.07**	68.63	**70.92**	**97.51**

iments on Qwen3-8B. The overall performance trends across all configurations are shown in Figure 4 and the detailed results for reasoning benchmarks are summarized in Table 3. More detailed results can be found in Appendix C.

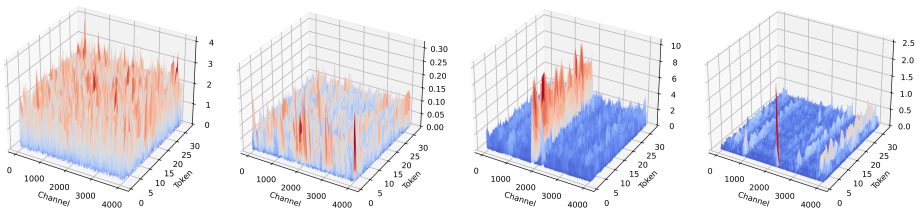
Non-Reasoning Tasks. As shown in Figure 4, under the `W4A8KV16` configuration, our method achieves near-lossless accuracy compared to BF16 baseline. As the quantization difficulty intensifies in the aggressive `W4A4KV16` and `W4A8KV4` regimes, rotation-based methods (e.g., SpinQuant, QuaRot) suffer from severe performance degradation while our method maintains a robust level of accuracy. This suggests that our block-wise affine transformation effectively mitigates the

Table 3: Performance comparison of various quantization methods on reasoning benchmarks across different bit-width configurations (e.g., W4A8KV16, W4A4KV16, W4A8KV8 and W4A8KV4).The recovery rate relative to the BF16 baseline is also provided and the best result in each case is marked in bold.

Bits	Method	GSM8K	MATH-500	AIME24	AIME25	GPQA-D	Avg.	*Recovery(%)*
BF16	–	95.15	96.87	71.46	63.12	58.13	76.95	100.00
W4A8KV16	RTN	93.71	95.53	64.58	55.00	54.39	72.64	93.64
	QuaRot	94.47	95.67	64.17	55.63	54.39	72.87	93.91
	SpinQuant	94.69	95.53	60.42	51.46	54.58	71.34	91.62
	BRQ	93.71	95.80	63.96	53.33	55.40	72.39	93.26
	FlatQuant	94.62	95.93	**69.17**	57.08	54.80	74.32	95.99
	SmoothQuant	**94.92**	96.27	65.62	56.04	54.80	73.53	94.80
	GPTQ	94.39	96.33	68.02	59.38	55.10	74.64	96.54
	BATQuant	94.84	**96.40**	68.33	**59.38**	**57.22**	**75.23**	**97.46**
W4A4KV16	RTN	93.10	94.53	53.33	47.08	49.80	67.57	86.06
	QuaRot	94.09	92.47	47.50	39.37	48.13	64.31	81.20
	SpinQuant	93.40	91.67	38.57	35.63	45.66	60.99	76.35
	BRQ	92.27	91.73	37.29	34.58	48.03	60.78	76.25
	FlatQuant	93.40	94.33	58.96	43.54	50.51	68.15	86.78
	SmoothQuant	94.69	95.33	60.71	47.29	52.42	70.09	89.60
	GPTQ	94.24	**95.73**	57.50	52.08	52.12	70.33	90.10
	BATQuant	**94.77**	95.60	**62.08**	**52.92**	**54.19**	**71.91**	**92.45**
W4A8KV8	RTN	93.78	95.00	60.21	54.79	53.54	71.46	91.96
	QuaRot	94.09	95.73	64.79	55.83	54.49	72.99	94.11
	SpinQuant	94.47	95.47	59.38	53.96	55.86	71.87	92.56
	BRQ	94.69	95.33	63.75	52.71	54.04	72.10	92.72
	FlatQuant	94.54	96.00	65.42	53.96	54.55	72.89	93.87
	SmoothQuant	94.39	96.13	66.04	54.79	54.29	73.13	94.21
	GPTQ	94.47	96.13	65.00	**57.08**	53.94	73.32	94.54
	BATQuant	**94.62**	**96.27**	**69.37**	55.21	**56.82**	**74.46**	**96.22**
W4A8KV4	RTN	92.12	91.13	43.54	38.75	46.97	62.50	78.80
	QuaRot	94.01	94.80	62.08	52.50	51.82	71.04	91.17
	SpinQuant	93.25	94.33	57.71	49.58	52.12	69.40	88.87
	BRQ	93.56	95.13	62.08	49.17	53.54	70.70	90.68
	FlatQuant	94.09	**95.40**	63.33	53.54	**54.95**	72.26	93.07
	SmoothQuant	93.03	92.73	46.67	40.33	49.19	63.39	81.46
	GPTQ	93.40	93.07	47.92	39.58	49.75	64.74	81.92
	BATQuant	**94.77**	95.27	**66.04**	**54.48**	54.24	**72.96**	**94.00**

distortion of activation distributions caused by extreme quantization, ensuring that fundamental linguistic patterns remain intact.

Reasoning Tasks. The disparity between BATQuant and baselines becomes even more pronounced on complex reasoning benchmarks requiring multi-step logical deduction and mathematical computation. As detailed in Table 3, reasoning tasks are inherently more sensitive to quantization noise due to the compounding effect of errors across long reasoning chains. In the `W4A8KV16` scenario, BATQuant achieves a recovery rate of 97.46%, surpassing GPTQ by a substantial margin of 0.92%. Notably, under `W4A4KV16` scenario, competing methods suffer from severe performance collapse on GSM8K and MATH-500, while BATQuant



(a) SpinQuant **(b)** FlatQuant **(c)** BRQ **(d)** BATQuant (Ours)

Fig. 5: Activation distributions of the `q_proj` module in layer 6 of Qwen3-8B with different quantization methods.

maintains a stable performance. In `W4A8KV8` and `W4A8KV4` scenarios, our method outperforms the strong baseline GPTQ and FlatQuant by 1.68% and 0.93%, respectively.

The consistent superiority of BATQuant across both multimodal tasks and complex linguistic reasoning underscores its remarkable cross-modality generalization. Our method maintains stable performance even under aggressive low-bit configurations where baselines fail. This broad effectiveness stems from the fundamental nature of our block-wise affine transformation, which dynamically aligns activation outliers and mitigates quantization noise at a granular level, independent of specific data modalities or task semantics.

Qualitative Results. To provide insights into the mechanism behind our performance gains, we visualize the activation distributions across different quantization methods in Figure 5. As shown in Figure 5a, rotation-based method (e.g., SpinQuant) tend to smooth the entire tensor. While it preserves the global energy, it may transfer energy from outlier-rich blocks to smoother blocks, amplifying quantization errors in these blocks. While FlatQuant (Figure 5b) effectively suppresses global energy, it fails to prevent this inter-block energy transfer. Furthermore, although BRQ (Figure 5c and Figure 2a) introduces block-wise rotation to smooth within local blocks, our visualization reveals that it often induces a bimodal distribution within quantization blocks. Our method (Figure 5d and Figure 2b) effectively prevents cross-block energy transfer while reshaping activations within blocks into a compact, unimodal distribution. More visualization results are provided in Appendix C.

4.3 Ablation Study

To validate the effectiveness of our core designs, we conduct ablation studies on both Qwen3-8B (LLM) and Qwen3-VL-8B-Instruct (MLLM) under the `W4A4KV16` configuration. Here, we first study the effect of block-wise affine transformation and block-wise learnable clipping.

Table 4: Ablation study of block-wise affine transformation and block-wise learnable clipping. We conduct the experiments under `W4A4KV16`.

Model	Components		Non-Reasoning Benchmarks					Avg.
	Block Trans	Block Clip	ARC-C	ARC-E	HellaSwag	PIQA	Winogrande	
Qwen3-8B	✓		53.16	76.36	71.02	74.27	**67.72**	68.51
		✓	52.35	77.44	**71.71**	**76.01**	63.69	68.24
	✓	✓	**53.33**	**77.53**	71.12	75.30	66.22	**68.70**
Model	**Components**		**Multimodal Benchmarks**					**Recovery**
	Block Trans	Block Clip	MME	OCRBench	DocVQA	RealWorldQA	VLMBlind	(%)
Qwen3-VL-8B -Instruct	✓		2235	861	**94.63**	67.19	69.99	96.18
		✓	2249	**865**	94.04	67.21	**70.28**	95.59
	✓	✓	**2360**	864	94.31	**67.32**	69.70	**96.43**

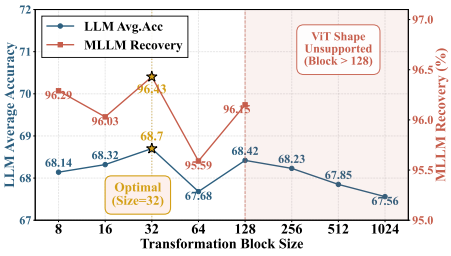


Fig. 6: Performance of Qwen3-8B (LLM) and Qwen3-VL-8B-Instruct (MLLM) with different transformation block sizes.

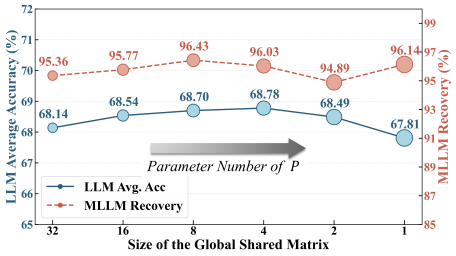


Fig. 7: Performance of Qwen3-8B (LLM) and Qwen3-VL-8B-Instruct (MLLM) with different sizes of the global shared matrix.

Effect of Block-wise Components. The baseline setting without block-wise affine transformation and block-wise learnable clipping refers to the use of *global-wise* counterparts. As shown in Table 4, replacing the global transformation with our block-wise variant yields significant improvements. For Qwen3-8B, applying the block-wise transformation improves the average accuracy from 68.24% to 68.70%. Similarly, for Qwen3-VL-8B-Instruct, it boosts the recovery rate from 95.59% to 96.43%. Applying block-wise clipping also provides competitive gains. For Qwen3-8B, the average accuracy is improved from 68.51% to 68.70%. For Qwen3-VL-8B-Instruct, the recovery rate is boosted from 96.18% to 96.43%. These confirm that using block-wise affine transformation and block-wise learnable clipping under MXFP quantization is crucial.

Block Size of Affine Matrix. BATQuant aligns the block size of affine transformation to the MXFP quantization granularity. To investigate the effect of transformation scope, we vary the size of the affine transformation $\mathbf{P}_i$ while keeping the MXFP quantization block size fixed at $g = 32$. As illustrated in Figure 6, for Qwen3-VL-8B-Instruct and Qwen3-8B, the best performance are both achieved when the transformation block size exactly matches the quantization block size ($g = 32$). This allows affine transformations to precisely reshape local

distributions, isolated from cross-block outlier interference. We can also observe that deviating from this alignment leads to performance degradation. When the block size of affine matrix is smaller than g (e.g., 16), the transformation scope is narrow to smooth outliers in quantization blocks. Additionally, distinct transformations lead to uneven energy (ℓ_2-norm) suppression within quantization blocks, creating imbalanced distributions and inducing new *local outliers*. When the block size of affine matrix is greater than g (e.g., 128), the transformation mixes elements across multiple quantization blocks. This transfers energy between blocks, which can increase quantization error. These findings suggest that strictly coupling the affine transformation granularity with the hardware quantization block size is an effective design choice.

Effect of GPK. To investigate the impact of Global and Private Kronecker (GPK) module, we analyze the size of the global shared matrix $\mathbf{A}$ (denoted as g_1). Recall that $g = g_1 \cdot g_2 = 32$; thus, varying g_1 inherently changes the capacity of both the shared global basis and the block-specific private components. We evaluate configurations with $g_1 \in \{1, 2, 4, 8, 16, 32\}$. The results are shown in Figure 7. Contrary to the intuition that increasing learnable parameters (i.e., decreasing g_1) monotonically improves performance, our experiments reveal a non-monotonic trend with an optimal point at $g_1 = 8$ or $g_1 = 4$. When g_1 is large (e.g., 16 or 32), the dimension of the private matrix $\mathbf{B}_i$ becomes small ($g_2 \leq 2$), severely limiting the ability of each block to adapt its local distribution independently and leading to a performance drop. Conversely, when g_1 is small (e.g., 1 or 2), the number of private parameters increases significantly, theoretically offering higher capacity. However, the search space is also expanded. The optimizer may struggle to converge to a robust solution without more calibration data or hyperparameter tuning, leading to sub-optimal performance or instability. Therefore, to strike an optimal balance between accuracy and efficiency, we recommend the configuration with $g_1 = 8$ as the default setting.

4.4 Inference Efficiency

To evaluate the practical deployment benefits of BATQuant, we conduct experiments on Qwen3-8B, measuring end-to-end latency, throughput, and peak memory usage under varying batch sizes.

End-to-end latency and throughput. Figure 8 reports both representative-layer GEMM throughput and end-to-end generation latency/speedup over BF16 on Qwen3-8B. The results shows that BATQuant achieve $\approx$ **4.7**$\times$ throughput over the BF16 baseline and approching the Ideal MXFP4 GEMMs. Advantages also carry over end-to-end latency, the speedup increases steadily as the workload grows, reaching **1.82**$\times$ at batch size 64.

Peak memory usage. Table 5 reports the peak memory with KV caches on Qwen3-8B under varying batch sizes. We compare the BF16 baseline, MR-GPTQ

Table 5: Peak memory usage with KV caches and saving factor on Qwen3-8B (Prefill=256, Decode=32).

bs	Peak Memory (GB)			Saving Factor (↑)	
	BF16	BATQuant	MR-GPTQ	BAT vs. BF16	MR vs. BF16
1	15.30	**5.85**	5.89	**2.62**×	2.60×
2	15.35	**5.95**	6.03	**2.58**×	2.55×
4	15.41	**6.10**	6.25	**2.53**×	2.47×
8	15.56	**6.38**	6.69	**2.44**×	2.33×
16	15.86	**6.99**	7.61	**2.27**×	2.08×
32	16.52	**8.31**	9.57	**1.99**×	1.73×
64	17.65	**10.56**	12.95	**1.67**×	1.36×

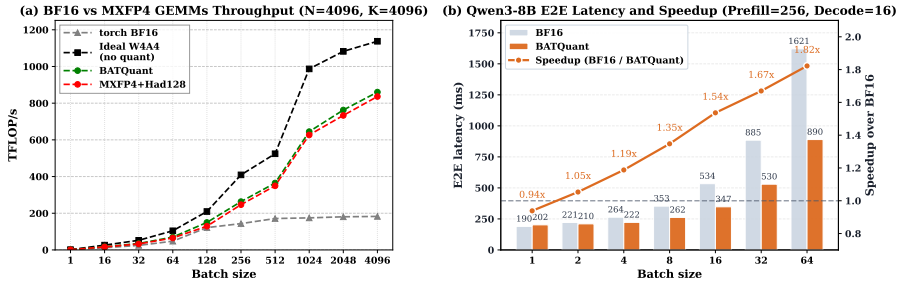


Fig. 8: (*Left*): GEMM throughput benchmark. (*Right*): E2E latency and speedup benchmark.

(MXFP4+Had128), and our method. BATQuant reduces memory over BF16 across batch sizes, achieving up to **2.62**× saving, while using slightly less memory than MR-GPTQ. These results indicate that BATQuant maintains the low memory footprint of MXFP4 quantization.

5 Conclusion

In this paper, we present BATQuant, a robust framework for outlier-resilient MXFP4 quantization that leverages learnable block-wise optimization. By restricting affine transformations to align strictly with hardware quantization granularity, our method effectively eliminates the cross-block energy transfer and bimodal distributions inherent in global rotation techniques. This targeted optimization, enhanced by efficient *Global and Private Kronecker* (GPK) decomposition and block-wise learnable clipping, ensures precise outlier suppression with minimal overhead. Extensive experiments on MLLMs and LLMs validate that BATQuant sets new state-of-the-art results, achieving **near-lossless** results under `W4A8KV16` and recovering up to **96.43%** of full-precision performance under aggressive `W4A4KV16` settings. We hope this work offers a practical solution for deploying large models on emerging microscaling architectures.

References

1. Advanced Micro Devices, Inc.: AMD CDNA™ 4 Architecture Whitepaper. White paper, Advanced Micro Devices, Inc. (2025)
2. Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R.K., Bai, Y., Baker, B., Bao, H., et al.: gpt-oss-120b & gpt-oss-20b model card. arXiv preprint arXiv:2508.10925 (2025)
3. Ashkboos, S., Mohtashami, A., Croci, M.L., Li, B., Cameron, P., Jaggi, M., Alistarh, D., Hoefler, T., Hensman, J.: Quarot: Outlier-free 4-bit inference in rotated llms. In: NeurIPS. pp. 100213–100240 (2024)
4. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
5. Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al.: Piqa: Reasoning about physical commonsense in natural language. In: AAAI. vol. 34, pp. 7432–7439 (2020)
6. Chen, M., Wu, M., Jin, H., Yuan, Z., Liu, J., Zhang, C., Li, Y., Huang, J., Ma, J., Xue, Z., et al.: Int vs fp: A comprehensive study of fine-grained low-bit quantization formats. arXiv preprint arXiv:2510.25602 (2025)
7. Choquette, J.: Nvidia hopper h100 gpu: Scaling performance. IEEE Micro **43**(3), 9–17 (2023)
8. Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., Tafjord, O.: Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv preprint arXiv:1803.05457 (2018)
9. Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al.: Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168 (2021)
10. Cook, J., Guo, J., Xiao, G., Lin, Y., Han, S.: Four over six: More accurate nvfp4 quantization with adaptive block scaling. arXiv preprint arXiv:2512.02010 (2025)
11. Egiazarian, V., Castro, R.L., Kuznedelev, D., Panferov, A., Kurtic, E., Pandit, S., Marques, A., Kurtz, M., Ashkboos, S., Hoefler, T., et al.: Bridging the gap between promise and performance for microscaling fp4 quantization. arXiv preprint arXiv:2509.23202 (2025)
12. Frantar, E., Ashkboos, S., Hoefler, T., Alistarh, D.: Gptq: Accurate post-training quantization for generative pre-trained transformers. arXiv preprint arXiv:2210.17323 (2022)
13. Fu, C., Chen, P., Shen, Y., Qin, Y., Zhang, M., Lin, X., Yang, J., Zheng, X., Li, K., Sun, X., et al.: Mme: A comprehensive evaluation benchmark for multimodal large language models. In: NeurIPS Datasets and Benchmarks Track (2025)
14. Hong, W., Yu, W., Gu, X., Wang, G., Gan, G., Tang, H., Cheng, J., Qi, J., Ji, J., Pan, L., et al.: Glm-4.5 v and glm-4.1 v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. arXiv preprint arXiv:2507.01006 (2025)
15. Hu, W., Zhang, Z., Zhang, H., Zhang, C., Guo, C., Feng, Y., Hu, T., Li, G., Hu, G., Wang, J., et al.: M2xfp: A metadata-augmented microscaling data format for efficient low-bit quantization. arXiv e-prints pp. arXiv–2601 (2026)
16. Hu, X., Chen, Z., Yang, D., Xu, Z., Xu, C., Yuan, Z., Zhou, S., Yu, J.: Moequant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance. arXiv preprint arXiv:2505.03804 (2025)
17. Huang, X., Liu, Z., Liu, S.Y., Cheng, K.T.: Rolora: Fine-tuning rotated outlier-free llms for effective weight-activation quantization. In: Findings of EMNLP. pp. 7563–7576 (2024)

18. Hudson, D.A., Manning, C.D.: Gqa: A new dataset for real-world visual reasoning and compositional question answering. In: CVPR. pp. 6700–6709 (2019)
19. Lee, J., Park, J., Cha, S., Cho, J., Sim, J.: Mx+: Pushing the limits of microscaling formats for efficient large language model serving. In: Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture. pp. 869–883 (2025)
20. LI, J., Beeching, E., Tunstall, L., Lipkin, B., Soletskyi, R., Huang, S.C., Rasul, K., Yu, L., Jiang, A., Shen, Z., Qin, Z., Dong, B., Zhou, L., Fleureau, Y., Lample, G., Polu, S.: Numinamath (2024)
21. Li, M., Lin, Y., Zhang, Z., Cai, T., Li, X., Guo, J., Xie, E., Meng, C., Zhu, J.Y., Han, S.: Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. arXiv preprint arXiv:2411.05007 (2024)
22. Li, S., Hu, Y., Ning, X., Liu, X., Hong, K., Jia, X., Li, X., Yan, Y., Ran, P., Dai, G., et al.: Mbq: Modality-balanced quantization for large vision-language models. In: CVPR. pp. 4167–4177 (2025)
23. Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., Cobbe, K.: Let's verify step by step. In: ICLR (2023)
24. Lin, H., Xu, H., Wu, Y., Cui, J., Zhang, Y., Mou, L., Song, L., Sun, Z., Wei, Y.: Duquant: Distributing outliers via dual transformation makes stronger quantized llms. In: NeurIPS. pp. 87766–87800 (2024)
25. Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.M., Wang, W.C., Xiao, G., Dang, X., Gan, C., Han, S.: Awq: Activation-aware weight quantization for on-device llm compression and acceleration. Proceedings of Machine Learning and Systems **6**, 87–100 (2024)
26. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: NeurIPS. pp. 34892–34916 (2023)
27. Liu, R., Sun, Y., Zhang, M., Bai, H., Yu, X., Yu, T., Yuan, C., Hou, L.: Quantization hurts reasoning? an empirical study on quantized reasoning models. arXiv preprint arXiv:2504.04823 (2025)
28. Liu, W., Meng, H., Luo, Y., Zhang, P., Ma, X.: Micromix: Efficient mixed-precision quantization with microscaling formats for large language models. arXiv preprint arXiv:2508.02343 (2025)
29. Liu, X., Xia, X., Zhang, M., Li, J.F., Yu, X., Shen, F., Su, X., Ng, S.K., Chua, T.S.: Freeact: Freeing activations for llm quantization. arXiv preprint arXiv:2603.01776 (2026)
30. Liu, X., Xia, X., Zhao, W., Zhang, M., Yu, X., Su, X., Yang, S., Ng, S.K., Chua, T.S.: L-mtp: Leap multi-token prediction beyond adjacent context for large language models. In: NeurIPS (2026)
31. Liu, Y., Li, Z., Huang, M., Yang, B., Yu, W., Li, C., Yin, X.C., Liu, C.L., Jin, L., Bai, X.: Ocrbench: on the hidden mystery of ocr in large multimodal models. Science China Information Sciences **67**(12), 220102 (2024)
32. Liu, Z., Zhao, C., Fedorov, I., Soran, B., Choudhary, D., Krishnamoorthi, R., Chandra, V., Tian, Y., Blankevoort, T.: Spinquant: Llm quantization with learned rotations. arXiv preprint arXiv:2405.16406 (2024)
33. Luo, R., Wang, L., He, W., Chen, L., Li, J., Xia, X.: Gui-r1: A generalist r1-style vision-language action model for gui agents. arXiv preprint arXiv:2504.10458 (2025)
34. Ma, Y., Li, H., Zheng, X., Ling, F., Xiao, X., Wang, R., Wen, S., Chao, F., Ji, R.: Affinequant: Affine transformation quantization for large language models. arXiv preprint arXiv:2403.12544 (2024)

35. Mathew, M., Karatzas, D., Jawahar, C.: Docvqa: A dataset for vqa on document images. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 2200–2209 (2021)
36. Meng, H., Luo, Y., Zhao, Y., Liu, W., Zhang, P., Ma, X.: Arcquant: Boosting nvfp4 quantization with augmented residual channels for llms. arXiv preprint arXiv:2601.07475 (2026)
37. Mishra, A., Stosic, D., Layton, S., Micikevicius, P.: Recipes for pre-training llms with mxfp8. arXiv preprint arXiv:2506.08027 (2025)
38. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019)
39. Qin, G., Li, Z., Chen, Z., Zhang, W., Kong, L., Zhang, Y.: Veq: Modality-adaptive quantization for moe vision-language models. arXiv preprint arXiv:2602.01037 (2026)
40. Rein, D., Hou, B.L., Stickland, A.C., Petty, J., Pang, R.Y., Dirani, J., Michael, J., Bowman, S.R.: Gpqa: A graduate-level google-proof q&a benchmark. In: COLM (2024)
41. Rouhani, B.D., Zhao, R., More, A., Hall, M., Khodamoradi, A., Deng, S., Choudhary, D., Cornea, M., Dellinger, E., Denolf, K., et al.: Microscaling data formats for deep learning. arXiv preprint arXiv:2310.10537 (2023)
42. Sakaguchi, K., Bras, R.L., Bhagavatula, C., Choi, Y.: Winogrande: An adversarial winograd schema challenge at scale. Communications of the ACM **64**(9), 99–106 (2021)
43. Shao, W., Chen, M., Zhang, Z., Xu, P., Zhao, L., Li, Z., Zhang, K., Gao, P., Qiao, Y., Luo, P.: Omniquant: Omnidirectionally calibrated quantization for large language models. arXiv preprint arXiv:2308.13137 (2023)
44. Shao, Y., Wang, P., Chen, Y., Xu, C., Wei, Z., Cheng, J.: Block rotation is all you need for mxfp4 quantization. arXiv preprint arXiv:2511.04214 (2025)
45. Sun, Y., Liu, R., Bai, H., Bao, H., Zhao, K., Li, Y., Hu, J., Yu, X., Hou, L., Yuan, C., et al.: Flatquant: Flatness matters for llm quantization. arXiv preprint arXiv:2410.09426 (2024)
46. Team, K., Du, A., Yin, B., Xing, B., Qu, B., Wang, B., Chen, C., Zhang, C., Du, C., Wei, C., et al.: Kimi-vl technical report. arXiv preprint arXiv:2504.07491 (2025)
47. Tirumala, A., Wong, R.: Nvidia blackwell platform: Advancing generative ai and accelerated computing. In: HCS. pp. 1–33 (2024)
48. Tseng, A., Chee, J., Sun, Q., Kuleshov, V., De Sa, C.: Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. ICML (2024)
49. Wang, H., Ma, S., Wei, F.: Bitnet v2: Native 4-bit activations with hadamard transformation for 1-bit llms. arXiv preprint arXiv:2504.18415 (2025)
50. Wang, W., Gao, Z., Gu, L., Pu, H., Cui, L., Wei, X., Liu, Z., Jing, L., Ye, S., Shao, J., et al.: Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. arXiv preprint arXiv:2508.18265 (2025)
51. Wang, W., Chen, W., Luo, Y., Long, Y., Lin, Z., Zhang, L., Lin, B., Cai, D., He, X.: Model compression and efficient inference for large language models: A survey. arXiv preprint arXiv:2402.09748 (2024)
52. Wei, X., Zhang, Y., Li, Y., Zhang, X., Gong, R., Guo, J., Liu, X.: Outlier suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling. In: EMNLP. pp. 1648–1665 (2023)

53. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations. pp. 38–45 (2020)
54. Wu, Z., Chen, X., Pan, Z., Liu, X., Liu, W., Dai, D., Gao, H., Ma, Y., Wu, C., Wang, B., et al.: Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. arXiv preprint arXiv:2412.10302 (2024)
55. xAI: Realworldqa: A benchmark for real-world spatial understanding (2024)
56. Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., Han, S.: Smoothquant: Accurate and efficient post-training quantization for large language models. In: ICML. pp. 38087–38099 (2023)
57. Xin, M., Priyadarshi, S., Xin, J., Kartal, B., Vavre, A., Thekkumpate, A.K., Chen, Z., Mahabaleshwarkar, A.S., Shahaf, I., Bercovich, A., et al.: Quantization-aware distillation for nvfp4 inference accuracy recovery. arXiv preprint arXiv:2601.20088 (2026)
58. Xu, M., Yin, W., Cai, D., Yi, R., Xu, D., Wang, Q., Wu, B., Zhao, Y., Yang, C., Wang, S., et al.: A survey of resource-efficient llm and multimodal foundation models. arXiv preprint arXiv:2401.08092 (2024)
59. Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al.: Qwen3 technical report. arXiv preprint arXiv:2505.09388 (2025)
60. Yu, J., Zhou, S., Yang, D., Li, S., Wang, S., Hu, X., Xu, C., Xu, Z., Shu, C., Yuan, Z.: Mquant: Unleashing the inference potential of multimodal large language models via static quantization. In: ACM MM. pp. 1783–1792 (2025)
61. Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., Choi, Y.: Hellaswag: Can a machine really finish your sentence? arXiv preprint arXiv:1905.07830 (2019)
62. Zeng, A., Lv, X., Zheng, Q., Hou, Z., Chen, B., Xie, C., Wang, C., Yin, D., Zeng, H., Zhang, J., et al.: Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. arXiv preprint arXiv:2508.06471 (2025)
63. Zhang, J., Wei, J., Zhang, P., Xu, X., Huang, H., Wang, H., Jiang, K., Chen, J., Zhu, J.: Sageattention3: Microscaling fp4 attention for inference and an exploration of 8-bit training. arXiv preprint arXiv:2505.11594 (2025)
64. Zhang, M., Li, J.F., Sun, Z., Bai, H., Zhen, H.L., Dong, Z., Yu, X.: Benchmarking post-training quantization of large language models under microscaling floating point formats. arXiv preprint arXiv:2601.09555 (2026)
65. Zhang, M., Li, J.F., Sun, Z., Liu, X., Dong, Z., Yu, X., Bai, H., Xia, X.: Quantclaw: Precision where it matters for openclaw. arXiv preprint arXiv:2604.22577 (2026)
66. Zhao, P., Zhen, H.L., Li, X., Bao, H., Lin, W., Yang, Z., Yu, Z., Wang, X., Yuan, M., Yu, X., et al.: Unleashing low-bit inference on ascend npus: A comprehensive evaluation of hifloat formats. arXiv preprint arXiv:2602.12635 (2026)
67. Zhu, X., Li, J., Liu, Y., Ma, C., Wang, W.: A survey on model compression for large language models. Transactions of the Association for Computational Linguistics **12**, 1556–1577 (2024)