SOT-BFP: Structure-aware Orthogonal Transformation for LLM Block Floating-Point Quantization

Xuandi Zeng,Yongqi Xu
School of Computer Science and Technology
Guangdong University of Technology
Guangzhou, 510006, China
3123004246@mail2.gdut.edu.cn,3121004930@mail2.gdut.edu.cn

Abstract. Block floating-point (BFP) quantization is an attractive solution for low-precision inference in large language models (LLMs) due to its wide dynamic range. However, its accuracy is fundamentally constrained by the shared exponent within each block: large within-block magnitude variation causes severe mantissa truncation for small values, leading to noticeable performance degradation. We propose SOT-BFP, a simple yet effective framework that mitigates this exponent-dominated truncation via structure-aware orthogonal transformation. Specifically, SOT-BFP first applies channel permutation to reorganize block composition, and then performs a block-wise orthogonal Hadamard transform to smooth within-block magnitude variation. This design constructs more favorable block structures for BFP quantization, while remaining exactly invertible in full precision and requiring no change to the BFP format or training pipeline. Experiments on widely used LLMs under post-training quantization show that SOT-BFP consistently improves downstream accuracy and reduces perplexity over standard BFP with negligible runtime overhead.

Keywords: Large language models, block floating-point quantization, low-bit inference, orthogonal transformation.

1 Introduction

Large language models have achieved remarkable success in applications such as dialogue, code generation, and complex reasoning. However, these capabilities come with substantial inference-time costs in memory, bandwidth, and computation, posing a major challenge for practical deployment [1, 2]. Low-precision quantization has therefore become a key technique for efficient LLM inference.

Among low-precision formats, block floating-point quantization offers an attractive trade-off between numerical range and efficiency [3, 4]. By allowing values within each block to share a common exponent, BFP reduces storage and bandwidth cost while preserving a wider dynamic range than standard integer formats. This makes BFP increasingly appealing for low-precision LLM deployment. However, existing BFP methods still suffer from noticeable accuracy degradation under low-bit

settings [5, 6]. This highlights the need to better understand and mitigate the quantization error caused by block-wise exponent sharing.

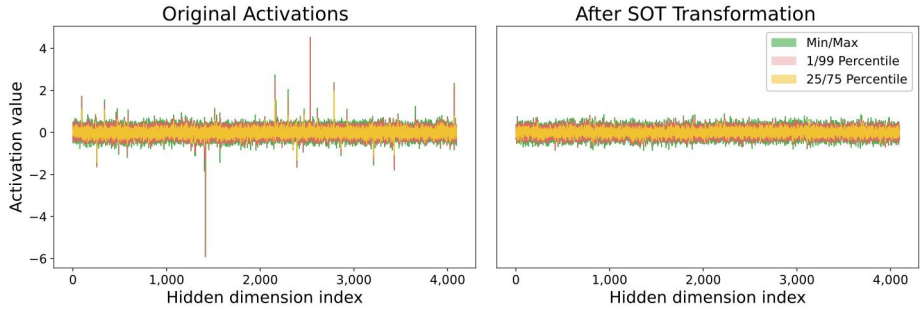

Fig. 1. Visualization of block-wise activation distributions before and after SOT Transform. SOT Transform reduces within-block magnitude variation, leading to smaller exponent gaps and less severe mantissa truncation under BFP quantization.

The main challenge of BFP lies in its shared-exponent mechanism. In each block, the exponent is determined by the largest-magnitude value, and all others must be right-shifted for alignment. When the intra-block dynamic range is large, small values suffer severe mantissa truncation. As shown in Fig. 1, LLM activations often contain outliers and uneven magnitudes across hidden dimensions, creating large exponent gaps within local blocks. Consequently, a few extreme values dominate exponent selection, while most smaller values lose substantial precision. We term this phenomenon exponent-dominated truncation error and identify it as a major source of accuracy degradation in low-bit BFP inference. This error is intrinsic to shared-exponent BFP and becomes more severe at lower bit widths.

Existing BFP methods do not explicitly target this error source. They typically preserve the original data distribution and apply exponent sharing directly, without reducing the block-wise exponent gap that governs truncation severity. This raises a key question: can exponent gaps be reduced before exponent sharing to systematically mitigate truncation error?

To answer this question, we propose SOT-BFP, a structure-aware orthogonal transformation framework for BFP quantization. The key idea is to introduce a BFP-oriented preconditioning step before quantization. Specifically, we first apply channel permutation to reorganize block composition, and then perform a block-wise orthogonal Hadamard transform to smooth within-block magnitude variation. The resulting transformation is exactly invertible in full precision, requires no change to the BFP format itself, and can be integrated into post-training quantization pipelines with negligible overhead.

Our main contributions are summarized as follows:

(1) We identify and formalize exponent-dominated truncation error as a critical yet underexplored accuracy bottleneck in low-bit BFP quantization.

(2) We propose SOT-BFP, a structure-aware orthogonal transformation framework that reduces block-wise exponent gaps through permutation-based block reorganization and block-wise Hadamard transformation.

(3) Experiments on OPT, LLaMA, and DeepSeek models show that SOT-BFP consistently improves accuracy and perplexity over representative baselines such as BIE [7] and SmoothQuant [8], while preserving the efficiency advantages of BFP.

2 Problem Formulation

2.1 Block Floating-Point Quantization

We first briefly review the formulation of block floating-point quantization. Let $X = \{x_i\}_{i=0}^{N-1}$ be a set of floating-point values, where each element is represented as

$$x_i = (-1)^{s_i} m_i 2^{e_i} \tag{1}$$

with s_i, m_i, and e_i denoting the sign, mantissa, and exponent, respectively.

In BFP representation, X is partitioned into K non-overlapping blocks $\{X^k\}_{k=1}^{K}$ of size B. All values within the same block share a common exponent, defined as

$$e^k = \max_{i \in X^k} e_i \tag{2}$$

This shared exponent is determined by the maximum magnitude value within the block. Given the shared exponent, each value within the block is quantized as

$$\hat{x}_i = (-1)^{s_i} m_i^{'} 2^{e^k},\ \mathrm{i} \in X^k \tag{3}$$

where the adjusted mantissa is obtained via

$$m_i^{'} = m_i \gg \left(e^k - e_i \right) \tag{4}$$

Here $\gg$ denotes the arithmetic right-shift operation. Sharing a block-wise exponent preserves a wider dynamic range than integer quantization; however, it also induces a block-dependent loss of effective mantissa precision through the exponent alignment in Eq. (4).

2.2 Exponent-Dominated Truncation Error

Building upon the BFP formulation in the previous section, we now analyze a key error mechanism specific to block floating-point quantization. In BFP, the shared exponent e^k within each block is determined by the maximum exponent, causing the quantization precision of all other values to be governed by the largest-magnitude element.

To characterize this effect, we define the intra-block exponent range as

$$\Delta e^k = \log_2 \frac{\max\limits_{i \in X^k} |x_i|}{\max\left(\min\limits_{i \in X^k} |x_i|,\ \varepsilon \right)} \tag{5}$$

where ε is a small positive constant to ensure numerical stability. This quantity measures the dynamic range within each block on a logarithmic scale.

A larger Δe^k implies a larger exponent gap $e^k - e_i$ for small-magnitude elements, leading to more aggressive right shifts in Eq. (4) and thus more severe truncation of mantissa bits.

We quantify the resulting error using the BFP reconstruction error:

$$\varepsilon_{BFP}(X) = \sum_{k=1}^{K} \sum_{i \in X^k} \left| x_i - \hat{x}_i \right|^2 \tag{6}$$

where $\hat{x}_i$ denotes the quantized value defined in Eq. (3).

Intuitively, $\varepsilon_{BFP}(X)$ increases with Δe^k, since the shared exponent is dominated by the largest-magnitude element in each block while the effective quantization resolution for smaller values becomes increasingly coarse. We refer to this phenomenon as exponent-dominated truncation error. Unlike rounding-error-centric views commonly used for integer quantization, this characterization pinpoints block-wise exponent domination as a primary driver of accuracy loss in low-bit BFP.

2.3 BFP-Oriented Transformation Principle

The above analysis reveals that the reconstruction error is fundamentally driven by the intra-block exponent range. Therefore, reducing Δe^k prior to exponent sharing becomes a principled strategy for mitigating exponent-dominated truncation error. To achieve this, we aim to reshape the activation distribution such that values within each block are more balanced in magnitude, thereby reducing exponent disparity and alleviating excessive right-shifting. Importantly, this process should preserve the original model computation and remain compatible with standard BFP quantization.

Based on these considerations, we consider transformations that reorganize and mix feature dimensions to reduce intra-block dynamic range. This principle directly motivates the structure-aware transformation introduced in the next section.

3 Method

3.1 Overview

We consider a linear layer $Y = XW$ in large language models, such as the projection and feed-forward layers in Transformer blocks. Here, $X \in \mathbb{R}^{M \times d}$ denotes the activation matrix, $W \in \mathbb{R}^{d \times n}$ the weight matrix, and $Y \in \mathbb{R}^{M \times n}$ the output. Our goal is to design a low-overhead pre-quantization transformation that reduces the intra-block exponent range and thereby mitigates the exponent-dominated truncation error.

An overview of the proposed SOT-BFP framework is shown in Fig. 2. The method first applies a channel permutation along the feature dimension, followed by a block-wise orthogonal Hadamard transform. The corresponding inverse transform is applied to the weights so that the original linear mapping is preserved in full precision.

We instantiate the abstract transformation $T(X; \pi, H)$ as a single orthogonal transform

$$G \triangleq PH \in \mathbb{R}^{d\times d} \tag{7}$$

where P is a channel permutation matrix and H is an orthogonal intra-block transform. The transformed activation is then obtained by applying G along the feature dimension.

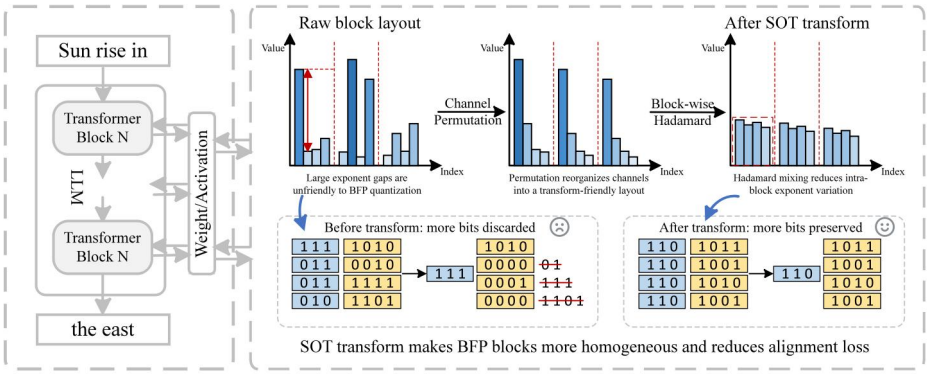

Fig. 2. Illustration of mantissa truncation before and after SOT transform. Before transform, large exponent gaps within a BFP block force small values to undergo excessive right-shifting during shared-exponent alignment, causing more mantissa bits to be discarded. After SOT transform, the values within the block become more homogeneous, reducing the exponent gap and preserving more effective bits during alignment.

3.2 Channel Permutation

We first introduce a channel permutation matrix $P \in \{0,1\}^{d\times d}$ satisfying $P^{\top}P = PP^{\top} = I$, which performs a global reordering along the feature dimension. The purpose of this permutation is not to directly reduce the scale variation within each block, but to mix channels with different typical magnitudes so that each contiguous block exhibits a heterogeneous scale pattern that is more amenable to the subsequent orthogonal transformation.

To construct this permutation, we first compute a per-channel magnitude statistic from a calibration set by averaging the absolute activation values,

$$a_j \triangleq \frac{1}{M}\sum_{i=1}^{M} |X_{ij}|, \quad j = 1, \ldots, \mathrm{d}. \tag{8}$$

These statistics characterize the typical scale of different channels. We then sort channels according to a_j and build the final permutation by interleaving channels from different magnitude ranges, rather than placing similarly scaled channels next to each other. In this way, channels with relatively large and small typical magnitudes are redistributed across the reordered feature dimension.

Applying the corresponding permutation matrix P to the activation yields

$$X' = XP \tag{9}$$

The key purpose of this reordering is to create a more favorable local structure for the subsequent block-wise Hadamard transform. After permutation, each contiguous BFP block tends to contain channels with more diverse scale characteristics, which makes the following orthogonal mixing more effective. In particular, although permutation alone usually yields limited gains, it rearranges the block structure in a way that enables the Hadamard transform to better redistribute energy within each block, alleviate local magnitude imbalance, and reduce exponent disparity. Therefore, channel permutation serves as a supportive preprocessing step: it does not independently resolve the quantization difficulty, but prepares a more transform-friendly block layout for the subsequent Hadamard transform.

3.3 Block-wise Hadamard Transform

After channel permutation, we apply an orthogonal intra-block transform to further reshape the distribution within each BFP block. While permutation reduces inter-channel scale disparity at a coarse level, individual blocks may still contain dominant elements that govern the shared exponent and induce excessive truncation. An additional intra-block transform redistributes such dominance across channels, reducing the block-wise exponent range in Eq. (5).

Let B denote the BFP block size along the feature dimension and assume $B|d$. For each token, the permuted feature vector is partitioned into $K = d/B$ non-overlapping groups of B consecutive channels. Within each group, we apply the same normalized Hadamard transform $H_B \in \mathbb{R}^{B\times B}$ satisfying $H_B H_B^\top = I$. Equivalently, the resulting intra-block transform can be written as

$$H = I_K \otimes H_B \in \mathbb{R}^{d\times d} \tag{10}$$

where $\otimes$ denotes the Kronecker product.

Importantly, the block partitioning is fully determined by the permutation P; the Hadamard transform operates strictly within each fixed block and does not alter block boundaries. This establishes a clear structural hierarchy: P performs a global channel reordering, while H applies a local orthogonal transform within each block.

The Hadamard transform is chosen for two practical reasons. First, it is orthogonal and therefore preserves the l_2 energy within each block, ensuring that the full-precision computation is numerically stable. Second, it is computationally efficient: multiplication by H_B can be implemented via the Fast Walsh--Hadamard Transform with $\mathcal{O}(B \log B)$ additions and negligible overhead.

Intuitively, the intra-block Hadamard transform reduces within-block dominance by redistributing energy across channels. A spiky coordinate is transformed into a more evenly spread pattern over the B channels, making the shared exponent less sensitive to a single element. Consequently, the exponent gap $e^k - e_i$ is reduced, directly mitigating exponent-dominated truncation under BFP quantization.

3.4 Invariance-Preserving Quantization

We design the pre-quantization transform in Eq. (7) to be invariance-preserving in full precision. Specifically, since $P^\top P = I$ and $H^\top H = I$, their product $G = PH$ is also orthogonal and satisfies $G^\top G = GG^\top = I$, implying $G^{-1} = G^\top$.

With this property, the original linear layer can be rewritten as

$$XW = X(GG^\top)W = (XG)(G^\top W) \tag{11}$$

which shows that right-multiplying activations by G and left-multiplying weights by $G^\top$ yields an exactly equivalent computation in full precision.

In the quantized setting, this equivalence no longer holds exactly because the BFP quantizer $Q_{\mathrm{BFP}}(\cdot)$ is nonlinear. We therefore apply BFP quantization after the change of basis,

$$\widehat{X} = Q_{\mathrm{BFP}}(XG), \qquad \widehat{W} = Q_{\mathrm{BFP}}(G^\top W) \tag{12}$$

and perform inference using $\widehat{Y} = \widehat{X}\widehat{W}$,Although $\widehat{Y}$only approximates Y, the transform G is explicitly chosen to make both XG and $G^\top W$ more amenable to low-bit BFP by reducing shared-exponent-induced truncation. In particular, P promotes within-block numerical homogeneity, while H further suppresses within-block dominance, jointly improving robustness under low-bit BFP quantization.

4 Evaluation

4.1 Experimental Setup

Baselines. All experiments are conducted in the post-training quantization (PTQ) setting. We compare our method against two representative approaches: (i) the fixed-point quantization method SmoothQuant with an INT8 configuration; and (ii) the block floating-point method BiE with a 4-bit mantissa.

Models and Datasets. We evaluate our method on three representative large language model families: OPT [9], LLaMA [10], and DeepSeek [11]. Specifically, we consider OPT-6.7B and OPT-13B, LLaMA2-7B and LLaMA2-13B, as well as DeepSeek-8B and DeepSeek-14B. We conduct experiments on natural language understanding, language modeling, and recall-based evaluation tasks. For natural language understanding, we report accuracy on ARC-Easy, ARC-Challenge, and HellaSwag. For language modeling, we report perplexity (PPL) on WikiText and LAMBADA. For recall-based evaluation, we report Recall@1 and Recall@2 on MuTual.

Implementation. Our method is applied prior to BFP quantization and consists of a channel permutation followed by a block-wise Hadamard transform. The inverse transform is applied to the weights to preserve the original full-precision linear mapping. The permutation order is estimated once from a small calibration set of M samples and remains fixed during inference. Unless otherwise stated, we adopt a W4A4 BFP configuration, where both weights and activations use 4-bit mantissas. A block size of 16 is used for both weights and activations, meaning that a single

exponent is shared across every 16 scalar values. All linear layers in the Transformer decoder, including attention and feed-forward projections, are quantized under this configuration. All models are implemented in PyTorch using HuggingFace Transformers, and all experiments are conducted on NVIDIA GeForce RTX~4090 GPUs under identical conditions. Unless otherwise stated, all measurements are conducted with an input sequence length of 2048 and a batch size of 32.

4.2 Main Results

We evaluate the effectiveness of the proposed method in mitigating exponent-dominated truncation error across both downstream natural language understanding tasks and language modeling benchmarks. We compare our approach with FP16 inference, SmoothQuant (SQ), the standard BFP format, and the representative BFP variant BIE. The results on natural language understanding and language modeling tasks are reported in Table 1 and Table 2, respectively.

Table 1. Comparison on natural language understanding tasks (% ↑).

Dataset	Model	FP	BFP	SQ	BIE	Ours
HellaSwag	OPT-6.7B	47.73	46.58	47.52	47.41	47.68
	OPT-13B	49.00	46.23	47.62	47.45	48.23
	LLaMA2-7B	50.08	47.62	48.98	48.93	49.21
	LLaMA2-13B	51.75	48.32	50.47	50.26	50.84
	DeepSeek-8B	49.50	47.92	48.74	48.53	49.12
	DeepSeek-14B	52.50	51.33	52.42	52.12	52.86
ARC-Challenge	OPT-6.7B	31.55	29.86	31.22	31.14	31.40
	OPT-13B	32.94	29.62	31.51	31.48	31.88
	LLaMA2-7B	43.43	41.26	42.40	42.34	42.52
	LLaMA2-13B	48.46	46.81	47.29	47.15	47.36
	DeepSeek-8B	41.96	40.68	41.30	41.25	41.73
	DeepSeek-14B	50.94	49.28	50.40	50.37	50.69

Table 2. Results on language modeling tasks (perplexity ↓).

Dataset	Model	FP	BFP	SQ	BIE	Ours
WikiText	OPT-6.7B	12.29	14.79	12.63	12.82	12.50
	OPT-13B	11.50	13.52	12.73	12.85	12.42
	LLaMA2-7B	8.71	9.47	9.29	9.32	9.15
	LLaMA2-13B	7.68	8.72	8.01	8.24	7.82
	DeepSeek-8B	13.74	15.06	14.21	14.37	13.85
	DeepSeek-14B	10.63	11.64	10.84	10.97	10.76
LAMBADA	OPT-6.7B	4.26	5.35	4.51	4.68	4.29
	OPT-13B	4.03	6.76	5.53	5.61	5.21
	LLaMA2-7B	3.33	4.21	3.53	3.65	3.48
	LLaMA2-13B	2.92	4.33	3.31	3.54	3.25
	DeepSeek-8B	4.46	5.47	4.62	4.96	4.43
	DeepSeek-14B	4.17	5.22	4.43	4.58	4.25

As shown in Table 1, our method consistently improves over vanilla BFP across all models and datasets, while achieving performance close to the FP16 baseline. Compared to BFP, the proposed method significantly narrows the accuracy gap, demonstrating the effectiveness of reducing intra-block exponent ranges. Moreover, our method is competitive with or slightly outperforms SmoothQuant and BIE in most settings, indicating that explicitly addressing exponent-sharing effects is crucial for preserving decision-level accuracy.

Table 2 reports the results on language modeling tasks. A similar trend is observed: our method consistently achieves lower perplexity than BFP and BIE, while remaining close to FP16 performance. Since perplexity is more sensitive to numerical errors, these results further confirm that reducing exponent disparity effectively mitigates truncation errors introduced by shared-exponent quantization.

We further observe that the performance gap between different quantization methods becomes more pronounced as model size increases. This can be attributed to increasingly skewed activation distributions in larger models, which amplify intra-block exponent disparities and exacerbate truncation errors. In such cases, the proposed method provides more substantial improvements, highlighting the importance of explicitly controlling exponent ranges in large-scale LLMs.

4.3 Recall-based Evaluation

To further evaluate the impact of quantization on ranking quality, we conduct experiments on the MuTual dataset, a retrieval-oriented dialogue understanding benchmark. Unlike classification accuracy, recall-based metrics are more sensitive to the relative ordering of model outputs, making them particularly suitable for assessing whether quantization preserves fine-grained score relationships. We report Recall@2 (R@2) in Table 3, which measures whether the correct response is ranked within the top-2 candidates.

Table 3. Recall@2 results on the MuTual dataset (% ↑).

Dataset	Model	FP	BFP	SQ	BIE	Ours
MuTual	OPT-6.7B	42.34	43.23	43.51	43.62	43.91
	OPT-13B	44.13	44.36	44.81	44.76	45.47
	LLaMA2-7B	42.78	43.00	46.35	46.39	46.70
	LLaMA2-13B	41.99	42.86	43.62	43.57	44.92
	DeepSeek-8B	43.10	43.52	43.59	43.54	43.71
	DeepSeek-14B	44.26	44.51	44.63	44.61	44.87

As shown in Table 3, our method consistently outperforms BFP and BIE and remains competitive with SmoothQuant across models. The improvement indicates better preservation of relative score ordering, which is crucial for ranking quality under quantization.

4.4 Ablation Study

4.4.1 Effect of Individual Components

We conduct an ablation study to examine the roles of channel permutation and block-wise Hadamard transform in SOT-BFP. Results on WikiText and ARC-Challenge are reported in Table 4.

Table 4. Ablation study under BFP quantization. WikiText reports PPL (↓) and ARC-Challenge reports Acc (↑).

Modules	OPT-13B		LLaMA2-13B	
	Wiki	ARC	Wiki	ARC
BFP	13.52	29.62	8.72	46.81
+ Channel Permutation	14.15	29.21	9.24	45.92
+ Block-wise Hadamard	12.98	30.96	8.54	47.05
+ SOT-BFP	12.42	31.88	7.82	47.36

Channel permutation alone slightly degrades performance on both OPT-13B and LLaMA2-13B, indicating that simple channel reordering is insufficient to reduce quantization error. In contrast, the block-wise Hadamard transform consistently improves perplexity and accuracy over the BFP baseline and serves as the primary source of performance gain. Combining both components achieves the best results, suggesting that channel permutation mainly plays a supportive role by creating a more transform-friendly channel layout, thereby further enhancing the effectiveness of the Hadamard transform in reducing intra-block heterogeneity.

4.4.2 Effect of Block Size

Table 5. Effect of block size on ARC-Challenge accuracy (%, ↑). "Before" and "After" denote the results before and after applying SOT-BFP, respectively.

Block Size	OPT-13B			LLaMa-13B		
	Acc. Before	Acc. After	Gain	Acc. Before	Acc. After	Gain
8	30.81	31.90	1.09	47.15	47.57	0.42
16	29.62	31.88	2.26	46.81	47.36	0.55
32	28.53	30.74	2.21	45.62	46.18	0.56
64	27.42	29.56	2.14	45.35	45.88	0.53
128	25.62	27.03	1.41	44.13	44.54	0.41

We further evaluate the sensitivity of SOT-BFP to different block sizes by varying the shared-exponent block size B from 8 to 128 while keeping all other settings unchanged. Results on ARC-Challenge are reported in Table 5. As the block size increases, the baseline BFP accuracy consistently decreases for both OPT-13B and LLaMA-13B, indicating that larger shared-exponent groups introduce more severe quantization error. In contrast, SOT-BFP consistently improves over the BFP baseline across all tested block sizes. The gain is relatively smaller at B=8, becomes more noticeable for B=16, 32, and 64, and decreases again at B=128. This suggests that smaller blocks already have limited intra-block heterogeneity, while very large blocks make the shared-exponent constraint too strong to be fully corrected by the proposed transformation.

4.4.3 sensitivity to calibration set size

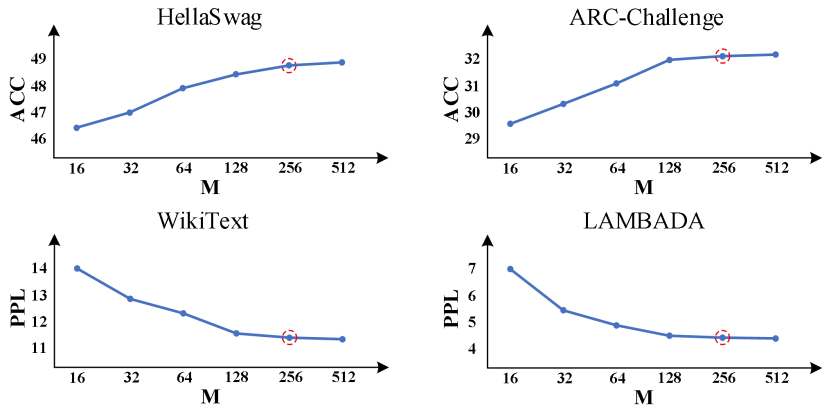

Fig. 3. Effect of calibration set size M on channel permutation effectiveness.

We vary the calibration set size $M \in \{16,32,64,128,256,512\}$ to study its impact on channel permutation. As shown in Fig. 3, increasing M improves the effectiveness of channel permutation and consistently reduces the final perplexity on OPT-13B. The improvement is more evident at small M, while it gradually saturates as M becomes larger. This result indicates that the proposed method only requires a modest calibration set to obtain a reliable permutation order.

4.5 End-to-End Inference Efficiency

To evaluate practical inference efficiency, we compare three BFP-based methods: vanilla BFP, BIE, and the proposed SOT-BFP. We measure efficiency using end-to-end decoding throughput in tokens per second (tokens/s), with sequence length set to 2048 and batch size set to 32. All results are normalized to the vanilla BFP baseline (BFP = 1), where higher values indicate better efficiency.

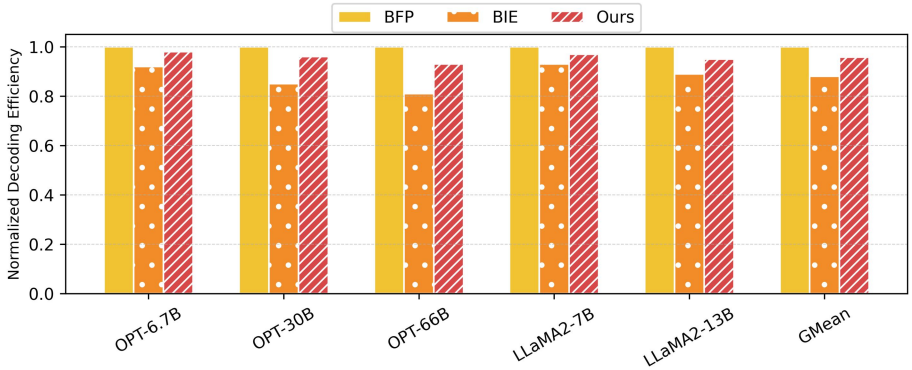

Fig. 4. Normalized decoding efficiency of BFP-based methods (BFP = 1). Higher is better. SOT-BFP incurs limited overhead and outperforms BIE.

As shown in Fig. 4, vanilla BFP achieves the highest decoding efficiency, while both BIE and SOT-BFP exhibit some efficiency degradation due to their additional processing steps. Compared with BIE, SOT-BFP consistently achieves higher decoding efficiency across different models and remains much closer to vanilla BFP. These results indicate that SOT-BFP improves quantization performance with only limited practical overhead, yielding a more favorable accuracy-efficiency trade-off than BIE.

From a method-structure perspective, the additional overhead of SOT-BFP mainly arises from extra memory access to intermediate activations rather than extra parameter storage. Channel permutation only changes the feature order, and the block-wise Hadamard transform is parameter-free, involving only structured add/sub operations. When these steps are executed separately, additional memory traffic may be introduced due to intermediate tensor reordering and read/write operations. In principle, optimized implementations such as kernel fusion or block-streaming could help reduce intermediate memory access by avoiding the explicit materialization of temporary tensors in off-chip memory.

5 Related Work

Block floating-point and related block-based numerical formats have attracted increasing attention for low-precision inference because they offer a favorable trade-off between dynamic range and efficiency [12, 13]. By allowing values within each block to share a common exponent, BFP reduces storage and arithmetic cost while preserving more dynamic range than standard integer formats. Recent work has revisited block-based quantization for low-bit inference and demonstrated its potential for efficient LLM deployment. More closely related to our setting, BIE improves BFP-style quantization through representation redesign and more flexible exponent allocation. These methods mainly enhance quantization through format-level modification. In contrast, our method does not change the BFP format itself, but instead introduces a pre-quantization transformation that directly reduces the block-wise exponent gap responsible for truncation under shared exponent selection.

Another related line of work improves quantization through output-preserving transformations. Rather than changing the model function, these methods map activations or weights into more quantization-friendly representations. QuaRot [14] shows that orthogonal rotations can suppress hidden-state outliers and improve low-bit inference, while SpinQuant [15] further demonstrates that the choice of transformation can significantly affect quantized performance. These works show that equivalent transformations are an effective tool for low-precision inference. However, their primary goal is to reduce outlier-induced quantization difficulty in integer or general low-bit settings, rather than to address the shared-exponent truncation mechanism specific to BFP. Our method is inspired by this transformation-based perspective, but specializes it to the BFP setting through structure-aware block reorganization and block-wise orthogonal transformation aimed at reducing exponent gaps before exponent sharing.

In contrast to prior work on BFP format design and generic transformation-based quantization, our method directly targets exponent-gap-induced truncation in shared-exponent BFP while leaving the original BFP format unchanged.

6 Conclusion

In this paper, we identify exponent-dominated truncation error as a key limitation of block floating-point quantization in large language models. To address this issue, we propose SOT-BFP, a structure-aware orthogonal transformation framework that reshapes activation distributions via channel permutation and block-wise Hadamard transforms, while preserving the original linear mapping through inverse weight transformation. Extensive experiments demonstrate that SOT-BFP consistently improves performance over vanilla BFP and strong baselines across multiple models and benchmarks, with minimal runtime overhead. Our results highlight the importance of aligning data distributions with numerical formats, and suggest that format-aware transformation is a promising direction for improving low-bit inference in large language models.

References

1. Jang, H., Song, J., Jung, J., Park, J., Kim, Y., Lee, J.: Smart-Infinity: Fast large language model training using near-storage processing on a real system. In: Proc. 2024 IEEE Int. Symp. High-Performance Computer Architecture (HPCA), pp. 345–360. IEEE, 2024.
2. Huang, Y., Wan, L.J., Ye, H., Jha, M., Wang, J., Li, Y., Zhang, X., Chen, D.: New solutions on LLM acceleration, optimization, and application. In: Proc. 61st ACM/IEEE Design Automation Conf. (DAC), pp. 1–4 (2024)
3. Song, Z. et al.: Computation error analysis of block floating point arithmetic oriented convolution neural network accelerator design. CoRR abs/1709.07776 (2017).
4. Zhang, H. et al.: A block-floating-point arithmetic based FPGA accelerator for convolutional neural networks. In: IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 1–5 (2019).
5. Lo, Y.-C., Lee, T.-K., Liu, R.-S.: Block and subword-scaling floating-point (BSFP): An efficient non-uniform quantization for low precision inference. In: The Eleventh International Conference on Learning Representations (ICLR) (2023).
6. Thompson, N., Fleming, M., Tang, B.J., Pastwa, A.M., Borge, N., Goehring, B.C., Das, S.: A model for estimating the economic costs of computer vision systems that use deep learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 23012–23018 (2024).
7. Zou, L., Zhao, W., Yin, S., Bai, C., Sun, Q., Yu, B.: BiE: Bi-exponent block floating-point for large language models quantization. In: International Conference on Machine Learning (ICML) (2024).
8. Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., Han, S.: SmoothQuant: Accurate and efficient post-training quantization for large language models. In: International Conference on Machine Learning (ICML), pp. 38087–38099. PMLR (2023)
9. Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., et al.: OPT: Open pre-trained transformer language models. arXiv preprint arXiv:2205.01068 (2022)

10. Touvron, H., et al.: LLaMA: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
11. DeepSeek-AI: DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv preprint arXiv:2501.12948 (2025)
12. Kohl, N., McCormick, S.F., Tamstorf, R.: Multigrid methods using block floating point arithmetic. SIAM Journal on Scientific Computing (2024) S202–S224.
13. Zhang, C., Cheng, J., Shumailov, I., Constantinides, G.A., Zhao, Y.: Revisiting block-based quantisation: What is important for sub-8-bit LLM inference? arXiv preprint arXiv:2310.05079 (2023).
14. 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: Advances in Neural Information Processing Systems (NeurIPS) (2024).
15. Liu, Z., Zhao, C., Fedorov, I., Soran, B., Choudhary, D., Krishnamoorthi, R., Chandra, V., Tian, Y., Blankevoort, T.: SpinQuant: LLM quantization with learned rotations. In: International Conference on Learning Representations (ICLR) (2024).