

Article

PBBQ: Plug-In Balanced Binary Quantization for LLMs

Zhangming Li ^{1,2} , Weifan Guan ^{1,2}, Zhengwei Chang ^{3,4}, Linghao Zhang ^{4,5} and Qinghao Hu ^{1,*}

¹ The Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China; lizhangming2023@ia.ac.cn (Z.L.); guanweifan2024@ia.ac.cn (W.G.)

² School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China

³ State Grid Sichuan Electric Power Company, Chengdu 610041, China; changzw@126.com

⁴ Power System Security and Operation Key Laboratory of Sichuan Province, Chengdu 610041, China; zlh.nju@gmail.com

⁵ State Grid Sichuan Electric Power Research Institute, Chengdu 610041, China

* Correspondence: huqinghao2014@ia.ac.cn

Abstract

In recent years, the expansion of large-model parameters has substantially increased storage and inference overhead. Consequently, post-training quantization has become a key technique for reducing model size and inference-time energy consumption. However, we observe that, under extremely low bit-width settings, mainstream error-compensation-based algorithms tend to overfit the calibration data. To mitigate this issue, we propose Plug-in Balanced Binary Quantization for LLMs (PBBQ), which reduces the excessive emphasis on subsequent channels via block-wise dropout and layer-wise reordering. PBBQ can be integrated into GPTQ-style frameworks and ultra-low-bit methods such as BiLLM and ARB-LLM. Experimental results show that PBBQ significantly improves the performance of multiple error-compensation quantization algorithms. When combined with the state-of-the-art methods BiLLM and ARB-LLM, the perplexity (ppl) on WikiText-2 is reduced by 21.46% (from 32.48 to 25.51) and 22.02% (from 16.44 to 12.82), respectively.

Keywords: large model compression; low-bit quantization; post-training quantization; error compensation quantization

1. Introduction

In recent years, rapid advances in deep learning and large-scale pre-trained models have delivered remarkable progress in natural language processing and computer vision. Foundation models such as GPT [1], LLaMA [2], and DeepSeek [3] have demonstrated strong general-purpose capabilities and are increasingly used as personal assistants in everyday scenarios. However, the explosive growth in parameter counts introduces substantial storage and compute burdens, which severely limits practical deployment and efficient inference.

This challenge is particularly critical for edge devices such as smartphones and personal PCs, where on-device LLMs are increasingly pursued for low latency, improved privacy, higher reliability (e.g., offline availability), and better personalization but are constrained by tight memory/energy budgets and limited thermal envelopes [4–6]. Recent real-world deployments further highlight these constraints: Apple reports an on-device foundation language model (on the order of a few billion parameters) designed for low-latency inference on consumer devices, alongside server models for more complex requests [7]; similarly, Google introduces Gemini models with a “Nano” variant to enable



Academic Editors: Nikos Temenos and Vasileios Ntinias

Received: 3 December 2025

Revised: 4 February 2026

Accepted: 4 February 2026

Published: 13 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

on-device generative AI use cases [8,9]. Beyond fully on-device inference, mobile edge intelligence (edge-cloud proximity serving) has also emerged as a practical pathway for latency-sensitive and cost-aware LLM applications, where computation can be offloaded to nearby edge servers while preserving user experience and certain privacy constraints [10]. In parallel, in cloud serving, LLM inference is widely limited by memory capacity and bandwidth and has become a major cost driver; reducing memory traffic and compute via quantization is therefore a direct lever to improve throughput and lower per-token cost [11].

Model quantization has emerged as an effective approach to compress neural networks, reducing model size and significantly lowering GPU/accelerator memory footprint, which is crucial for both edge deployment and cost-efficient cloud inference [12,13]. Contemporary quantization methods for LLMs are commonly categorized into quantization-aware training (QAT) and post-training quantization (PTQ). QAT incorporates quantization into training/fine-tuning and can deliver stronger low-bit accuracy, but it is often expensive or infeasible for large proprietary LLMs. In contrast, PTQ avoids backpropagation and enables fast, practical deployment using only a small calibration set, making it attractive for rapid iteration and real-world system integration [13,14]. For example, ZeroQuant demonstrates an efficient PTQ pipeline for large Transformer models to reduce the serving cost [14], while SmoothQuant shows that training-free PTQ can enable accurate and hardware-friendly low-bit inference by addressing activation outliers, achieving strong accuracy under efficient INT8 deployment [15]. Given the depth and scale of modern LLMs, PTQ is especially appealing in time- and resource-constrained scenarios, and it also serves as a key enabler for pushing toward ultra-low-bit regimes (e.g., 2/1-bit) when training access is limited.

Although previous post-training quantization methods have achieved success in 8-bit and 4-bit quantization, the continuous expansion of large language models requires more radical quantization schemes. Neural network binarization (reducing the weight bitwidth to 1 bit) is a promising approach. However, the current state-of-the-art post-training quantization methods for ultra-low-bit large language models still rely mainly on error compensation, and such methods often suffer from poor generalization and mediocre performance on non-calibration datasets. To tackle this problem, we conduct a series of empirical studies and obtain two key observations from the experimental results shown in Figure 1:

- Existing error-compensation-based quantization methods typically process and compensate channels in ascending order of their in-channel indices. As a result, each layer tends to overemphasize in-channel dimensions with larger indices. Moreover, since the Hessian matrix of the next layer is computed based on the output of the previous layer, the bias introduced by this unbalanced compensation accumulates across layers and ultimately leads to large errors in the final model output.
- After quantization, the weight matrices of large language models often exhibit a pattern in which the later in-channels have weights with larger magnitudes. This overemphasis on the subsequent channels causes the model to focus excessively on certain in-channel dimensions, which deviates from the distribution of the original full-precision weights.

Inspired by the above observations, we propose a plug-and-play post-training quantization method for large language models, termed Plug-in Balanced Binary Quantization (PBBQ), which incorporates two core designs to achieve high-accuracy weight binarization:

- First, to alleviate the tendency for later dimensions to receive disproportionately high compensation, we introduce a block-wise dropout strategy that discards smaller compensation values. Concretely, the dropout ratio is increased with the distance

between blocks, thereby preventing excessive compensation on the subsequent in-channel dimensions.

- Second, to make the quantized weight distribution closer to that of the original pre-quantized weights, we reorder the columns of the weight matrix according to their magnitudes prior to quantization. The corresponding permutation is recorded and then inverted after quantization, ensuring that the most compensated dimensions within each layer are aligned with those having larger original weight amplitudes.

Extensive experiments show that PBBQ has achieved state-of-the-art performance on various evaluation metrics for multiple large language model families. For example, combined with the mainstream SOTA methods BiLLM and ARB-LLM, the perplexity (ppl) on Wikitext2 is reduced by 21.46% (from 32.48 to 25.51) and 22.02% (from 16.44 to 12.82), respectively.

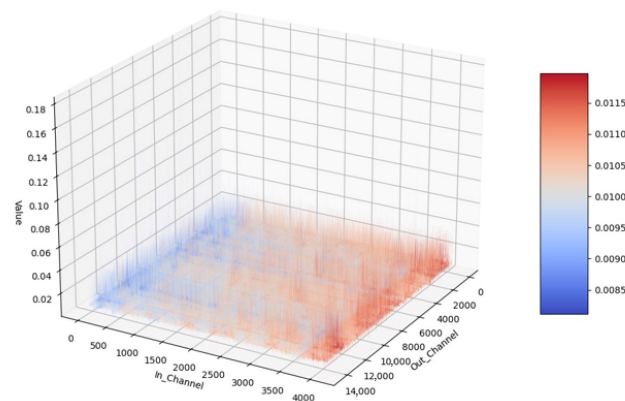


Figure 1. For the LLaMA-2-7B model quantized with an error-compensation-based PTQ algorithm, the more frequent compensation applied to later blocks ultimately leads to larger weight magnitudes in the corresponding quantized channels.

2. Related Work

Quantization converts weights and/or activations from floating-point (e.g., FP16/BF16) to low-bit fixed-point representations to reduce memory footprint and improve inference efficiency. While early quantization research mainly targeted convolutional networks [16–18], large language models (LLMs) exhibit distinct weight/activation distributions and extreme scale, motivating dedicated LLM quantization methods.

LLM quantization is commonly divided into quantization-aware training (QAT) and post-training quantization (PTQ). QAT typically delivers stronger low-bit accuracy but requires fine-tuning, which is expensive at LLM scale. LLM-QAT [19] proposes a data-free QAT strategy by distilling from model generations and also quantizing the KV cache; however, it still involves iterative optimization and is less lightweight than pure PTQ for rapid deployment.

PTQ avoids backpropagation and is therefore widely adopted for LLMs. PTQ can be further categorized into weight-only quantization and activation–weight joint quantization. Weight-only PTQ mainly reduces model memory and bandwidth and can often reach lower weight bit-widths. Joint quantization can leverage low-bit compute kernels (commonly INT8), but pushing activations to very low bit-widths often introduces non-negligible accuracy loss.

LLM.int8 [20] enables robust INT8 inference by handling activation outliers via mixed-precision decomposition, achieving strong accuracy in the 8-bit regime. Nevertheless, this line of work primarily targets INT8 and does not directly address the severe degradation observed when pushing toward ultra-low-bit weights (e.g., 2/1-bit) where compensation bias and distribution mismatch become dominant.

GPTQ [21] is a representative weight-only PTQ approach that uses approximate second-order information and block-wise processing with error compensation, achieving strong results at 3–4 bits. However, as bit-width decreases, reconstruction/compensation methods become more sensitive to the calibration set and may generalize poorly when error propagation accumulates across blocks and layers. AWQ [22] improves robustness by leveraging activation statistics to identify salient channels and applying channel-wise scaling to reduce quantization loss, but it is mainly designed for INT3/INT4 rather than 1-bit binarization. SpQR [23] stores outlier weights at higher bit-widths, quantizes the remaining weights to low bit-widths using fine-grained group quantization, and employs a sparse storage format for outliers. QuIP [24] improves very-low-bit quantization via incoherence processing and adaptive rounding, but it relies on additional transformations and processing steps beyond standard block-wise compensation. Norm Tweaking [25] rectifies the quantized activation distribution by adjusting normalization layers, improving generalization; however, it modifies model parameters and is not tailored to the order-induced bias in sequential compensation.

Binarization (1-bit weights) provides maximal compression but is substantially more fragile than 2–4 bit quantization. PB-LLM [26] introduces partial binarization by keeping a small subset of critical weights at higher precision to preserve reasoning, at the cost of mixed precision. BiLLM [27] advances 1-bit PTQ with structured saliency selection and binary residual approximation, improving accuracy under 1-bit settings. ARB-LLM [28] further refines binarization through iterative parameter updates and enhanced partitioning, achieving stronger performance. Despite these advances, many ultra-low-bit PTQ methods still depend on GPTQ-style sequential error compensation. In this setting, processing blocks in a fixed in-channel order can induce order-dependent compensation bias, where later blocks receive repeated residual updates, leading to distribution drift and potential overfitting to calibration data, a failure mode that becomes more pronounced at 2/1-bit.

PBBQ is a plug-and-play module designed specifically for GPTQ-style block-wise error-compensation pipelines. It is complementary to saliency selection, residual modeling, and iterative refinement used in prior binary PTQ methods [26–28]. PBBQ targets the above order-induced bias and cross-layer error accumulation via two orthogonal designs: (i) layer-wise in-channel reordering to align more-compensated positions with larger-magnitude channels; and (ii) block-wise dropout on propagated compensation to suppress excessive accumulation on later blocks, thereby improving robustness and generalization in ultra-low-bit regimes.

3. Method

In this section, we begin by presenting the preliminaries. Then we provide a detailed discussion on the PBBQ method, which consists of two parts: layer-wise reorder and block-wise dropout, as illustrated in Figure 2. Finally, we describe the complete PBBQ algorithm within the overall pipeline of PBBQ.

3.1. Preliminaries

We consider a linear layer with weight matrix $W \in \mathbb{R}^{n \times m}$ and input (calibration) activations $X \in \mathbb{R}^{m \times r}$ collected from a small calibration set. The full-precision output is

$$Y = WX. \quad (1)$$

Post-training quantization aims to construct a low-bit weight matrix B (e.g., 3/2/1-bit) such that the quantized output $Y_q = BX$ approximates Y on the calibration data. GPTQ-type methods formulate this as a layer-wise reconstruction problem with a Hessian-weighted objective:

$$\min_B \|(W - B)X\|_H^2 \quad \text{with} \quad H = 2XX^\top + \lambda I. \quad (2)$$

where $H \in \mathbb{R}^{m \times m}$ is an approximation of the input Hessian and λ is a small regularizer.

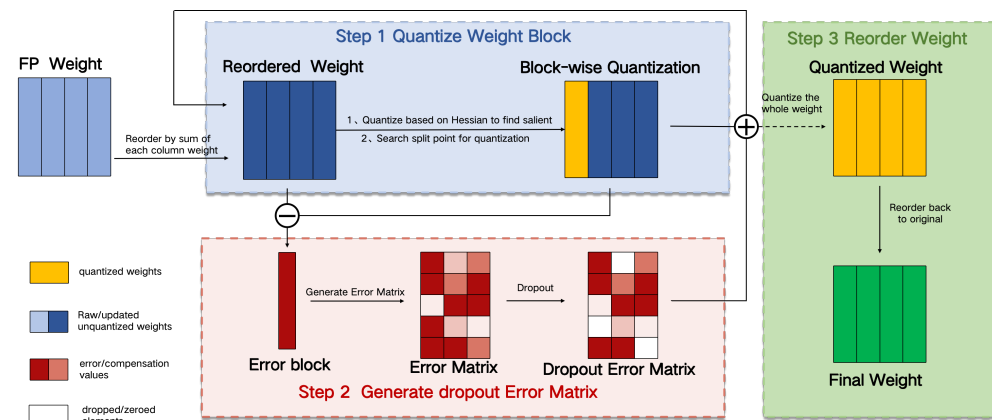


Figure 2. Overall pipeline of PBBQ. Step 1: reorder the weight matrix and perform GPTQ-style block-wise quantization. Step 2: form the block-wise error matrix and apply value-based dropout before propagating compensation. The red heatmap represents the error/compensation values (darker red indicates larger magnitude), and white cells indicate dropped/zeroed elements after dropout. Step 3: reorder the quantized weights back to the original layout to obtain the final binarized weights.

To make the optimization tractable for large W , GPTQ partitions the in-channel dimension into blocks of size β (e.g., $\beta = 128$) and processes them sequentially from the first to the last block. For the b -th block $W^b = W_{:,b:b+\beta}$, the algorithm:

- quantizes W^b to a low-bit block B^b .
- computes the residual $E^b = (W^b - B^b)/H^{-1}$.
- propagates E^b to the unquantized blocks by updating their full-precision weights as $W \leftarrow W - E^b H^{-1}$, so that their future quantization can compensate the current error.

3.2. Layer-Wise Reorder

When word embeddings are first fed into the model, Layer Normalization (LN) ensures that their overall scale is comparable and that the distribution across dimensions is relatively uniform. However, after GPTQ-style weight compensation is applied in each layer, the later in-channel dimensions are repeatedly compensated and amplified, causing these dimensions to dominate the layer's output. This bias is then propagated to the $(n + 1)$ -th layer through the Hessian-based computation in that layer, since the same ascending in-channel order (from 0 to 4096 in LLaMA-2-7B) is used again.

To mitigate this overemphasis on high-index dimensions and make the quantized weights closer to the original distribution, we introduce a layer-wise reorder operator that explicitly permutes the in-channels before quantization. Given a weight matrix $W \in \mathbb{R}^{n \times m}$, we first compute a column importance score for each in-channel j :

$$s_j = \sum_{i=1}^n |W_{ij}|, \quad j = 1, \dots, m. \quad (3)$$

Let π be a permutation of $\{1, \dots, m\}$ that sorts these scores in ascending order:

$$s_{\pi(1)} \leq s_{\pi(2)} \leq \dots \leq s_{\pi(m)}. \quad (4)$$

We define the reordered weight matrix by permuting the in-channel (column) indices:

$$\hat{W}_{:,k} = W_{:,\pi(k)}, \quad k = 1, \dots, m. \quad (5)$$

The same permutation is applied to the Hessian matrix.

$$\hat{H} = P_{\pi}^{\top} H P_{\pi}. \quad (6)$$

where P_{π} is the permutation matrix associated with π .

We then perform block-wise quantization and error compensation on the reordered pair $(\hat{W}, \hat{H})$ using the GPTQ + PBBQ procedure. After obtaining the quantized weights $\hat{B}$, we restore the original in-channel order by applying the inverse permutation:

$$B_{:, \pi(k)} = \hat{B}_{:, k}, \quad k = 1, \dots, m. \quad (7)$$

In this way, channels that originally have larger weight magnitudes are more likely to be placed later in the quantization order and thus receive more compensation, making the compensated (and binarized) weights better aligned with the full-precision distribution.

3.3. Block-Wise Dropout

Since the blocks are processed in ascending order, later blocks are repeatedly updated with the accumulated residuals from all previous ones, making GPTQ-style methods prone to overfitting the specific calibration distribution, as shown in Figure 1. To alleviate this, PBBQ preserves the overall GPTQ compensation framework but modifies how the residuals are propagated in the block-wise scheme. Specifically, the in-channel dimension is partitioned into blocks of size β and processed sequentially. For the b -th block, after obtaining the binarized (or low-bit) weights $B_{:, b:b+\beta}$, we compute the residual:

$$E = \frac{W_{:, b:b+\beta} - B_{:, b:b+\beta}}{H_{bb:b+\beta b+\beta}^c}. \quad (8)$$

where H^c is the Cholesky factor used in the main procedure. In standard GPTQ-style compensation, this residual is then propagated to all later blocks to update their full-precision weights.

In PBBQ, we keep the same compensation structure but insert a block-wise dropout on the propagated error. Specifically, for the $(i+1)$ -th future block ($i \geq 0$), we first compute the propagated compensation

$$\tilde{E}^{(i)} = E H_{b:b+i\beta, b+(i+1)\beta}^c. \quad (9)$$

We then construct a binary mask $M^{(i)} \in \{0, 1\}^{n \times \beta}$ that drops the smallest p_i fraction of elements in $\tilde{E}^{(i)}$:

$$\tilde{E}^{(i) \text{ drop}} = M^{(i)} \odot \tilde{E}^{(i)}, \quad p_i = p \cdot \gamma_i. \quad (10)$$

where $\odot$ denotes element-wise multiplication, p is the base prune ratio, and γ_i is calculated as follows:

$$\gamma_i = \frac{i}{N_u} \quad (11)$$

where N_u is the number of subsequent blocks to be updated and i indicates the block position (farther blocks get larger dropout). The weight update for the $(i+1)$ -th block becomes

$$W_{:, b+i\beta: b+(i+1)\beta} \leftarrow W_{:, b+\beta:} - \tilde{E}^{(i) \text{ drop}}. \quad (12)$$

This dropout suppresses excessive error accumulation on later blocks, prevents their weights from becoming disproportionately large, and thus alleviates the over-attention to high-index in-channel dimensions. In practice, we find that setting the base prune ratio p such that the effective dropout rate p_i lies around 20–25% yields a good balance between error compensation and generalization. Figure 3 is our schematic diagram.

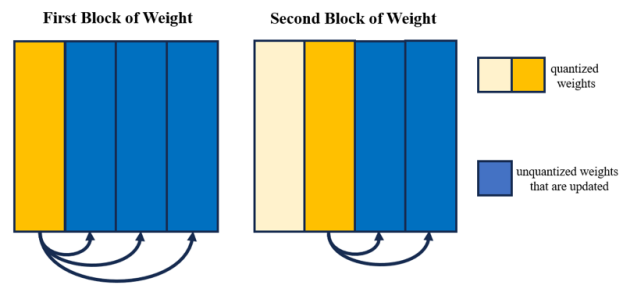


Figure 3. The schematic diagram of block by block compensation shows that the current compensation method may result in duplicate compensation for later blocks. In order to reduce this bias, we sort the compensation values from small to large and then dropout the topk.

3.4. Pipeline of PBBQ

The PBBQ algorithm follows the general paradigm of existing quantization error compensation methods and can be used as a plug-in module on top of mainstream GPTQ-based algorithms. Its pseudocode is given in Algorithm 1. Specifically, lines 1–5 implement the layer-wise reorder step, lines 6–16 follow the original BiLLM procedure, and lines 17–22 realize the proposed block-wise dropout mechanism.

Algorithm 1 Main Framework of PBBQ

func PBBQ($W, X, \beta, \lambda, p, f$)

Input: $W \in \mathbb{R}^{n \times m}$ — weight matrix
 $X \in \mathbb{R}^{m \times r}$ — calibration data, dtype is float32
 β — block size
 λ — hessian regularizer
 p — prune ratio
 f — Whether to reorder

Output: B — binarized weights

```

1:  $H \leftarrow 2XX^T$  // dtype of H is always float32
2: if  $f == \text{True}$  then
3:    $W = W \cdot \text{reorder}(\text{dim} = 1)$ 
4:    $H = H \cdot \text{reorder}(\text{dim} = 0) \cdot \text{reorder}(\text{dim} = 1)$ 
5: end if
6:  $H^c = \text{Cholesky}((H + \lambda I)^{-1})$ 
7:  $B = 0_{n \times m}$ 
8: while  $b = 0, \beta, 2\beta, \dots, N$  do
9:    $W^b = W_{:,b:b+\beta}$ 
10:   $\text{rows} = \text{salient}(W^b, H^c)$ 
11:   $\tilde{B}_1 = \text{res\_approximation}(W^b_{:,j \in \{\text{rows}\}})$ 
12:   $p^* = \text{seg\_search}(W^b_{i,j \notin \{\text{rows}\}})$ 
13:   $\tilde{B}_2 = \text{binary}(W^b_{|w_{i,j}| \leq p^*, j \notin \{\text{rows}\}})$ 
14:   $\tilde{B}_3 = \text{binary}(W^b_{|w_{i,j}| > p^*, j \notin \{\text{rows}\}})$ 
15:   $B_{:,b:b+\beta} = \tilde{B}_1 + \tilde{B}_2 + \tilde{B}_3$ 
16:   $E = (W_{:,b:b+\beta} - B_{:,b:b+\beta}) / H^c_{bb:b+\beta:b+\beta}$ 
17:  while  $i = 0, 1, 2, \dots, N-b$  do
18:     $\tilde{E} = E \cdot H^c_{b:b+i\beta, b+(i+1)\beta}$ 
19:     $\text{indices} = \text{topk}(\tilde{E}.\text{all}(), \tilde{E}.\text{numel}() \cdot p \cdot (i/N - b), \text{largest} = \text{False})$ 
20:     $\tilde{E}.\text{all}()[\text{indices}] = 0$ 
21:     $W_{:,b+i\beta:b+(i+1)\beta} = W_{:,b:b+\beta} - \tilde{E}$ 
22:  end while
23: end while
24: if  $f == \text{True}$  then
25:    $W = W \cdot \text{reorder}(\text{dim} = 1)$ 
26: end if
27: return  $B$ 

```

4. Experiments

4.1. Datasets

In terms of datasets, we evaluate the effectiveness of our method on multiple language modeling benchmarks, including WikiText-2 [29], C4 [30], PTB [31], and MMLU [32]. WikiText2 is an open-source benchmark dataset for language modeling, constructed by high-quality Wikipedia articles. It comprises roughly 2.09 million training tokens with a vocabulary of approximately 33,278 words. C4 (Colossal Clean Crawled Corpus) is a large-scale English text dataset built by filtering and processing Common Crawl web data. PTB (Penn Treebank) is a classic language modeling corpus (Wall Street Journal text) and is commonly used with the standard preprocessing that yields about 1M tokens and a 10K-word vocabulary. MMLU (Massive Multitask Language Understanding) is a multi-domain multiple-choice benchmark designed to evaluate broad factual knowledge and problem-solving ability. It contains 15,908 questions spanning 57 subjects across STEM, humanities, and social sciences, covering difficulty levels from elementary to professional; each question provides four answer options.

4.2. Evaluation Protocol

Regarding evaluation metrics, we follow BiLLM and adopt perplexity (PPL) as the primary measure. Perplexity is widely used in language modeling to assess how well a model predicts the next token in a sequence: it is essentially the exponential of the average negative log-likelihood. A lower PPL indicates that the model assigns higher probability to the ground-truth tokens and thus reflects better predictive performance.

Importantly, perplexity is highly sensitive to quantization noise, making it an effective indicator of accuracy degradation introduced by low-bit quantization. Therefore, comparing PPL before and after quantization provides a clear and quantitative way to evaluate the impact of our method on model quality.

In addition, on the MMLU benchmark we report accuracy, defined as the fraction of questions answered correctly. Specifically, we evaluate all models using the EleutherAI lm-evaluation-harness (lm-eval) codebase and report multiple-choice accuracy on MMLU.

4.3. Setup

The experiments of PBBQ are primarily conducted using the PyTorch v2.0.1 [33] and Hugging Face library v0.16.4 [34]. For the calibration data, we follow GPTQ and related work, using 128 random sequences of length 2048 sampled from the C4 dataset, which mainly consists of general web text. For comparison, we evaluate against representative large-model quantization methods, including RTN, GPTQ, SliM-LLM [35], PB-LLM, BiLLM, and ARB-LLM. Since our approach focuses on memory-aligned quantization, we do not compare with sparse-based methods (e.g., SpQR). SpQR achieves near-lossless 3–4 bit weight compression by explicitly isolating outlier weights and storing them in higher precision with a sparse outlier component, while quantizing the remaining weights to low bit-width. In contrast, PBBQ is designed for ultra-low-bit dense binarization on top of GPTQ-style error compensation. In addition, SpQR drop weights in the final model and maintain any explicit outlier set and mixed-precision outlier storage, while PBBQ dropout is applied only to the propagated compensation updates during reconstruction. Following the unified evaluation protocol in [36], we further evaluate the MMLU benchmark and report multiple-choice accuracy, defined as the fraction of questions answered correctly.

4.4. Experimental Results

To evaluate the generality of the proposed method, we conduct experiments on large language models at multiple scales, including Llama-2-7B/13B/70B [37] and Llama3-8B/70B [38], thereby demonstrating its applicability across a wide range of model sizes.

As shown in Table 1, augmenting GPTQ, BiLLM, and ARB-LLM with PBBQ consistently reduces perplexity at the same bit-width across LLaMA-2-7B/13B/70B. Under the 3-bit setting, the improvement is relatively modest but stable (GPTQ 6.44 vs. GPTQ + PBBQ 6.33). In contrast, at 2-bit (GPTQ 60.45 vs. GPTQ + PBBQ 47.69) and especially in the 1-bit regime, the gains become much more pronounced, indicating that block-wise dropout and layer-wise reordering effectively suppress the over-compensation of later channels and better preserve the original weight distribution. In particular, PBBQ achieves substantial PPL reductions on top of strong baselines such as BiLLM (32.48 vs. 25.51) and ARB-LLM (16.44 vs. 12.82), demonstrating that our method is complementary to existing error-compensation and saliency-based binarization strategies. A similar trend can be observed in Llama-3 in Table 2. For both 3-bit and 2-bit GPTQ and 1-bit BiLLM/ARB-LLM, PBBQ further narrows the gap to full precision. These results demonstrate that (i) PBBQ is architecture-agnostic across Llama-2 and Llama-3 families, (ii) it remains effective from 7B/8B to 70B scales, and (iii) it can be used as a plug-and-play module to systematically enhance a wide range of GPTQ-style post-training quantization algorithms in the ultra-low-bit setting.

Table 1. Perplexity comparison of RTN, GPTQ, PB-LLM, BiLLM, ARB-LLM, and our proposed method on Llama-2. The table reports perplexity on the WikiText-2 dataset across different model scales.

Method	Block Size	Weight Bits	7B	13B	70B
Full Precision	-	16.00	5.47	4.88	3.32
RTN	-	3.00	542.80	10.68	7.53
GPTQ	128	3.00	6.44	5.46	3.88
GPTQ + PBBQ	128	3.00	6.33	5.42	3.87
RTN	-	2.00	17,788.93	51,145.61	26,066.13
QuIP	-	2.00	39.73	13.48	6.64
Slim-LLM	-	2.16	16.01	9.41	6.28
GPTQ	128	2.00	60.45	24.48	8.18
GPTQ + PBBQ	128	2.00	47.69	20.88	8.07
RTN	-	1.00	157,058.34	47,902.32	160,389.91
PB-LLM	128	1.70	66.41	236.40	28.37
BiLLM	128	1.08	32.48	21.77	13.32
BiLLM + PBBQ	128	1.08	25.51	20.27	12.61
ARB-LLM _{RC}	128	1.08	16.44	11.85	6.16
ARB-LLM _{RC} + PBBQ	128	1.08	12.82	11.18	5.83

We also measure quantization overhead: averaged over five runs on LLaMA-2-7B, quantizing one layer takes 1158.57 s with ARB-LLM_{RC} and 1160.45 s with ARB-LLM_{RC} + PBBQ (i.e., +1.88 s, +0.16%). This confirms that PBBQ introduces negligible extra quantization cost, as it only adds a lightweight column permutation and simple value-based masking during offline reconstruction. Moreover, PBBQ incurs no inference overhead: the reorder operation is fully inverted after quantization, so the final quantized weights are stored in the original layout, and inference uses the same GEMM as BiLLM; dropout is applied only to the propagated compensation during offline quantization.

Table 2. Perplexity comparison of RTN, GPTQ, PB-LLM, BiLLM, ARB-LLM, and our proposed method on Llama-3. The columns report perplexity on the WikiText2 dataset for different model sizes.

Method	Block Size	Weight Bits	8B	70B
Full Precision	-	16.00	6.14	2.86
RTN	-	3.00	2194.98	13,592.69
GPTQ	128	3.00	18.68	6.65
GPTQ + PBBQ	128	3.00	8.47	6.39
RTN	-	2.00	1,335,816.13	481,927.66
GPTQ	128	2.00	1480.43	82.23
GPTQ + PBBQ	128	2.00	1371.76	70.68
RTN	-	1.00	1,353,698.38	375,658.34
PB-LLM	128	1.70	73.08	22.96
BiLLM	128	1.06	58.98	66.30
BiLLM + PBBQ	128	1.06	51.99	60.43
ARB-LLM _{RC}	128	1.06	27.42	11.10
ARB-LLM _{RC} + PBBQ	128	1.06	25.96	7.94

To verify whether the perplexity improvements at 1-bit translate to preserved reasoning performance, we additionally evaluate LLaMA-2-7B under the 1.08-bit setting on the MMLU benchmark using lm-eval and report multiple-choice accuracy. As shown in Table 3, PBBQ consistently improves the downstream accuracy on top of strong 1-bit baselines: BiLLM improves from 22.3 to 23.0, and ARB-LLM_{RC} improves from 23.7 to 25.3 under the same 1.08-bit setting, indicating that the gains in perplexity also translate to better preserved downstream performance.

Table 3. Zero-shot MMLU accuracy (%) on LLaMA-2-7B under 1.08-bit quantization.

Method	Block Size	Weight Bits	7B
Full Precision	-	16.00	41.8
BiLLM	128	1.08	22.3
BiLLM + PBBQ	128	1.08	23.0
ARB-LLM _{RC}	128	1.08	23.7
ARB-LLM _{RC} + PBBQ	128	1.08	25.3

We further evaluate robustness to calibration data distribution shifts while keeping the standard GPTQ calibration protocol fixed, as shown in Table 4. Using C4, WikiText-2, or PTB as calibration corpora, the WikiText-2 perplexity of ARB-LLM_{RC} + PBBQ changes only slightly (12.82/12.67/13.10, respectively), indicating that PBBQ is robust to moderate shifts in calibration data distribution rather than overfitting to a specific calibration set.

Table 4. Effect of different calibration corpora on WikiText-2 perplexity.

Method	Calibration	Block Size	Weight Bits	7B
ARB-LLM _{RC} + PBBQ	C4	128	1.08	12.82
ARB-LLM _{RC} + PBBQ	WikiText-2	128	1.08	12.67
ARB-LLM _{RC} + PBBQ	PTB	128	1.08	13.10

Lastly, we study the sensitivity of the base dropout ratio p under LLaMA-2-7B (ARB-LLM_{RC} + PBBQ, block size 128, 1.08-bit), as shown in Figure 4. Varying p shows a clear optimum: base dropout in [0.20, 0.60] consistently outperforms the ARB-LLM_{RC} baseline,

while the best perplexity is achieved in the $[0.20, 0.25]$ range (12.85–12.92), with a near-identical minimum at $p = 0.22$ (12.82). In contrast, both larger settings (≥ 0.30) and a smaller setting (0.10) substantially degrade performance, suggesting that PBBQ is stable within $[0.20, 0.25]$ but sensitive to overly strong or overly weak dropout.

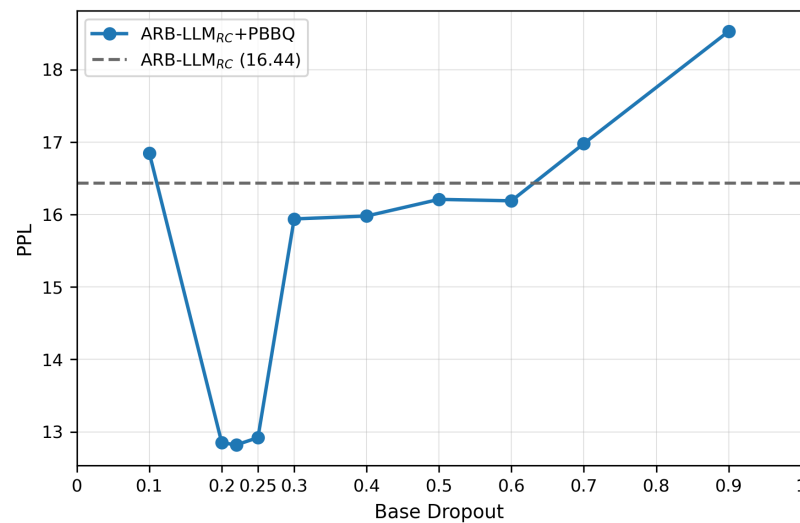


Figure 4. Sensitivity of the base dropout ratio p on LLaMA-2-7B under ARB-LLM_{RC} + PBBQ.

4.5. Ablation Study

To assess the contribution of each component in PBBQ, we conduct an ablation study on Llama-2-7B with 1.08-bit quantization using ARB-LLM_{RC} as the baseline. As reported in Table 5, the baseline ARB-LLM_{RC} obtains 16.44 PPL on WikiText2 and 20.12 PPL on C4.

Adding only the layer-wise reorder module (w/o dropout) already yields a clear improvement, reducing perplexity to 13.53 (WikiText2) and 16.98 (C4). Introducing only block-wise dropout (w/o reorder) brings a larger gain, further lowering perplexity to 13.01 and 15.84, respectively. When both block-wise dropout and layer-wise reorder are enabled, PBBQ achieves the best results, reaching 12.82 PPL on WikiText2 and 15.50 PPL on C4. These results confirm that each module is effective on its own and that their combination is complementary, jointly enhancing the robustness and accuracy of 1-bit error-compensation quantization.

In addition, we evaluated several alternative column-ordering strategies, including Hessian-diagonal-based sorting (ranking columns by $\text{diag}(XX^T)$ as well as L2-norm, max-value, and variance-based sorting on LLaMA-2-7B using ARB-LLM_{RC} + PBBQ, and report perplexity (PPL) on WikiText-2 as the evaluation metric. As shown in Table 6, our method (L1 norm) obtains the best results. Therefore, we keep the proposed column-magnitude-based (L1 norm) reordering, which is both effective and lightweight in our setting.

At last, we compare several column-ordering strategies on LLaMA-2-7B using ARB-LLM_{RC} + PBBQ (block size 128, 1.08-bit) and report WikiText-2 perplexity in Table 6. L1-magnitude-based ordering achieves the best result and is also lightweight; therefore, we adopt it as the default.

Table 5. Ablation study on Llama-2-7B.

Algorithm	Total Bits	Dropout	Reorder	Wikitext2 ↓	C4 ↓
ARB-LLM _{RC}	1.08	✗	✗	16.44	20.12
ARB-LLM _{RC} + PBBQ	1.08	✗	✓	13.53	16.98
ARB-LLM _{RC} + PBBQ	1.08	✓	✗	13.01	15.84
ARB-LLM _{RC} + PBBQ	1.08	✓	✓	12.82	15.50

Table 6. Comparison of different column reordering criteria.

Method	Score	Block Size	Weight Bits	7B
ARB-LLM _{RC} + PBBQ	L1	128	1.08	12.82
ARB-LLM _{RC} + PBBQ	Hessian-diag	128	1.08	16.57
ARB-LLM _{RC} + PBBQ	L2	128	1.08	78.06
ARB-LLM _{RC} + PBBQ	Max	128	1.08	16.93
ARB-LLM _{RC} + PBBQ	Variance	128	1.08	61.99

5. Conclusions

In this paper, we propose PBBQ, a plug-and-play post-training weight-quantization module for ultra-low-bit compression of text-only LLMs. PBBQ combines (i) layer-wise in-channel reordering to better align compensation with the original weight distribution and (ii) block-wise dropout to suppress over-compensation on later channels in GPTQ-style error propagation. We validate PBBQ on the LLaMA-2/3 families under PTQ settings and evaluate model quality using perplexity on WikiText2 and C4 (and downstream task accuracy). Across model scales up to 70B, attaching PBBQ to representative GPTQ-style baselines (e.g., GPTQ, BiLLM, and ARB-LLM) consistently reduces perplexity at the same bit-width, with the most pronounced gains in the 1-bit regime. In terms of efficiency, our binarization setting yields up to ~93% weight-memory reduction compared with FP16 storage, while maintaining competitive accuracy. In future work, we plan to explore dynamic mixed-precision quantization and hardware-adaptive optimizations to further improve performance and deployment efficiency. We also plan to generalize the plug-in design to broader quantization regimes, including activation quantization and multimodal models.

Author Contributions: Conceptualization, Z.L. and W.G.; methodology, Z.L. and Q.H.; software, Z.L.; validation, Z.L.; formal analysis, Z.L.; investigation, W.G.; resources, Z.C. and Q.H.; data curation, L.Z.; writing—original draft preparation, Z.L.; writing—review and editing, Z.L.; visualization, Z.L.; supervision, Z.C. and Q.H.; project administration, L.Z.; and funding acquisition, L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Science and Technology Project of State Grid Corporation of China (5700-202426249A-1-1-ZN).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. WikiText-2 and C4 can be found at their corresponding project websites.

Acknowledgments: We appreciate the fund from the Science and Technology Project of State Grid Corporation of China (5700-202426249A-1-1-ZN).

Conflicts of Interest: Author Zhengwei Chang was employed by the company State Grid Sichuan Electric Power Company. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LLM	Large language model
PTQ	Post-training quantization
PBBQ	Plug-in Balanced Binary Quantization

References

- Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. *Improving Language Understanding by Generative Pre-Training*; OpenAI Technical Report; OpenAI: San Francisco, CA, USA, 2018.
- Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. Llama: Open and efficient foundation language models. *arXiv* **2023**, arXiv:2302.13971.
- Bi, X.; Chen, D.; Chen, G.; Chen, S.; Dai, D.; Deng, C.; Ding, H.; Dong, K.; Du, Q.; Fu, Z.; et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv* **2024**, arXiv:2401.02954.
- Xu, J.; Li, Z.; Chen, W.; Wang, Q.; Gao, X.; Cai, Q.; Ling, Z. On-Device Language Models: A Comprehensive Review. *arXiv* **2024**, arXiv:2409.00088.
- Wang, R.; Gao, Z.; Zhang, L.; Yue, S.; Gao, Z. Empowering large language models to edge intelligence: A survey of edge efficient LLMs and techniques. *Comput. Sci. Rev.* **2025**, *57*, 100755.
- Cai, G.; Tian, R.; Yang, L.; Jia, Y.; Li, L.; Wang, J. Efficient Inference for Edge Large Language Models. *Tsinghua Sci. Technol.* **2026**, *31*, 1365–1380.
- Gunter, T.; Wang, Z.; Wang, C.; Pang, R.; Narayanan, A.; Zhang, A.; Zhang, B.; Chen, C.; Chiu, C.-C.; Qiu, D.; et al. Apple Intelligence Foundation Language Models. *arXiv* **2024**, arXiv:2407.21075.
- GOOGLE. Introducing Gemini: Our Largest and Most Capable AI model. Google Blog. 2023. Available online: <https://blog.google/innovation-and-ai/technology/ai/google-gemini-ai/> (accessed on 6 December 2023).
- GOOGLE. Gemini Nano is Now Available on Android via Experimental Access. Android Developers Blog. 2024. Available online: <https://android-developers.googleblog.com/2024/10/gemini-nano-experimental-access-available-on-android.html> (accessed on 1 October 2024).
- Qu, G.; Chen, Q.; Wei, W.; Lin, Z.; Chen, X.; Huang, K. Mobile Edge Intelligence for Large Language Models. *arXiv* **2024**, arXiv:2407.18921.
- Davies, M.; Crago, N.; Sankaralingam, K.; Kozyrakis, C. Efficient LLM Inference: Bandwidth, Compute, Synchronization, and Capacity are all you need. *arXiv* **2025**, arXiv:2507.14397.
- Zhu, X.; Li, J.; Liu, Y.; Ma, C.; Wang, W. A Survey on Model Compression for Large Language Models. In *Transactions of the Association for Computational Linguistics (TACL)*; MIT Press: Cambridge, MA, USA, 2024.
- Gong, R.; Ding, Y.; Wang, Z.; Lv, C.; Zheng, X.; Du, J.; Qin, H.; Guo, J.; Magno, M.; Liu, X. A Survey of Low-bit Large Language Models: Basics, Systems, and Algorithms. *arXiv* **2024**, arXiv:2409.16694.
- Yao, Z.; Aminabadi, R.Y.; Zhang, M.; Wu, X.; Li, C.; He, Y. ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS 2022)*, New Orleans, LA, USA, 28 November–9 December 2022.
- Xiao, G.; Lin, J.; Seznec, M.; Wu, H.; Demouth, J.; Han, S. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. In *Proceedings of the International Conference on Machine Learning (ICML 2023)*, Honolulu, HI, USA, 23–29 July 2023.
- Qiu, J.; Wang, J.; Yao, S.; Guo, K.; Li, B.; Zhou, E.; Yu, J.; Tang, T.; Xu, N.; Song, S.; et al. Going deeper with embedded fpga platform for convolutional neural network. In *FPGA '16: Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*; Chen, D., Greene, J.W., Eds.; ACM: New York, NY, USA, 2016; pp. 26–35. <https://doi.org/10.1145/2847263.2847265>.
- Lin, D.; Talathi, S.; Annapureddy, S. Fixed point quantization of deep convolutional networks. In *Proceedings of Machine Learning Research: Volume 48 Proceedings of The 33rd International Conference on Machine Learning*; Balcan, M.F., Weinberger, K.Q., Eds.; PMLR: New York, NY, USA, 2016; pp. 2849–2858.
- Courbariaux, M.; Bengio, Y.; David, J. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proceedings of the Advances in Neural Information Processing Systems 28*; Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., Eds.; Montreal, QC, Canada, 7–12 December 2015; pp. 3123–3131.
- Liu, Z.; Oguz, B.; Zhao, C.; Chang, E.; Stock, P.; Mehdad, Y.; Shi, Y.; Krishnamoorthi, R.; Chandra, V. LLM-QAT: Data-free quantization aware training for large language models. In *Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand, 11–16 August 2024.
- Dettmers, T.; Lewis, M.; Belkada, Y.; Zettlemoyer, L. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS 2022)*, New Orleans, LA, USA, 28 November–9 December 2022.
- Frantar, E.; Ashkboos, S.; Hoefler, T.; Alistarh, D. GPTQ: Accurate posttraining quantization for generative pre-trained transformers. In *Proceedings of the International Conference on Learning Representations (ICLR 2023)*, Kigali, Rwanda, 1–5 May 2023.
- 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 LLM compression and acceleration. In *Proceedings of the Machine Learning and Systems (MLSys 2024)*, Santa Clara, CA, USA, 13–16 May 2024.

23. Dettmers, T.; Svirschevski, R.; Egiazarian, V.; Kuznedelev, D.; Frantar, E.; Ashkboos, S.; Borzunov, A.; Hoefler, T.; Alistarh, D. SpQR: A Sparse-Quantized Representation for Near-Lossless LLM Weight Compression. In Proceedings of the International Conference on Learning Representations (ICLR 2024), Vienna, Austria, 7–11 May 2024.
24. Chee, J.; Cai, Y.; Kuleshov, V.; De Sa, C. M. QuIP: 2-Bit Quantization of Large Language Models With Guarantees. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS 2023), New Orleans, LA, USA, 10–16 December 2023.
25. Li, L.; Li, Q.; Zhang, B.; Chu, X. Norm tweaking: High-performance low-bit quantization of large language models. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2024), Vancouver, BC, Canada, 20–27 February 2024.
26. Yuan, Z.; Shang, Y.; Dong, Z. Pb-llm: Partially binarized large language models. In Proceedings of the International Conference on Learning Representations (ICLR 2024), Vienna Austria, 7–11 May 2024.
27. Huang, W.; Liu, Y.; Qin, H.; Li, Y.; Zhang, S.; Liu, X.; Magno, M.; Qi, X. Billm: Pushing the limit of post-training quantization for llms. In Proceedings of the 41st International Conference on Machine Learning (ICML 2024), Vienna, Austria, 21–27 July 2024.
28. Li, Z.; Yan, X.; Zhang, T.; Qin, H.; Xie, D.; Tian, J.; Shi, Z.; Kong, L.; Zhang, Y.; Yang, X. Arb-llm: Alternating refined binarizations for large language models. In Proceedings of the International Conference on Learning Representations (ICLR 2025), Singapore, 24–28 April 2025.
29. Merity, S.; Xiong, C.; Bradbury, J.; Socher, R. Pointer sentinel mixture models. In Proceedings of the International Conference on Learning Representations (ICLR 2017), Toulon, France, 24–26 April 2017.
30. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.
31. Marcus, M.; Kim, G.; Marcinkiewicz, M.A.; MacIntyre, R.; Bies, A.; Ferguson, M.; Katz, K.; Schasberger, B. The penn treebank: Annotating predicate argument structure. In Proceedings of the Human Language Technology: Proceedings of a Workshop, Plainsboro, NJ, USA, 8–11 March 1994.
32. Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; Steinhardt, J. Measuring massive multitask language understanding. In Proceedings of the International Conference on Learning Representations (ICLR 2021), Virtual, 3–7 May 2021.
33. 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 *Advances in Neural Information Processing Systems*; ACM: New York, NY, USA, 2019; Volume 32.
34. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. Huggingface’s transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP 2020 Demos), Virtual, 16–20 November 2020.
35. Huang, W.; Qin, H.; Liu, Y.; Li, Y.; Liu, Q.; Liu, X.; Benini, L.; Magno, M.; Zhang, S.; Qi, X. Slim-LLM: Salience-driven mixed-precision quantization for large language models. In Proceedings of the 42nd International Conference on Machine Learning (ICML 2025), Vancouver, BC, Canada, 13–19 July 2025.
36. Zhao, J.; Wang, M.; Zhang, M.; Shang, Y.; Liu, X.; Wang, Y.; Zhang, M.; Nie, L. Benchmarking post-training quantization in llms: Comprehensive taxonomy, unified evaluation, and comparative analysis. *arXiv* **2025**, arXiv:2502.13178.
37. Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*; Meta AI Research Publication: Menlo Park, CA, USA, 2023.
38. Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. *The Llama 3 Herd of Models*; Meta AI Research Publication: Menlo Park, CA, USA, 2024.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.