All for 1-Bit: Towards Genuine 1-Bit Post-Training Quantization for LLMs

Zhixiong Zhao[1,2*†] , **Zukang Xu**[1*] , **Guangyu Sun**[2] , **Lifeng Liu**[2‡] , **Dawei Yang**[1‡]

[1]Houmo AI
[2]School of Integrated Circuits, Peking University
Correspondence: dawei.yang@houmo.ai

Abstract

Large language models (LLMs) have achieved remarkable progress, yet their massive storage and memory-bandwidth demands still hinder efficient deployment. Weight binarization is a promising solution, but existing binarization-based post-training quantization (PTQ) methods usually far exceed the nominal 1-bit storage target due to hidden overhead. To address this gap, we propose **A**ll **f**or **1**-Bit (**AF1**), a genuine 1-bit PTQ framework for LLMs. AF1 comprises two complementary components: (1) Null-space-Aware Binary Factorization (NABF) for improving binary reconstruction through Hessian-aware surrogate reparameterization, null-space-aware binary factorization, and scale-only global reconstruction; and (2) Hierarchical Shapley Allocation (HiSA) for assigning structural capacity using hierarchical Shapley sensitivity. Together, they preserve model accuracy under a strict 1.0 effective-BPW budget for target linear weights in the PTQ setting. Experiments on LLaMA, Qwen, and Gemma families show that AF1 consistently outperforms existing binarization-based PTQ methods in perplexity and zero-shot accuracy. Compared with BF16, AF1 achieves an average **2.5**× inference speedup and over **90%** memory reduction across evaluated models, providing a practical path toward deployable genuine 1-bit compression for LLMs. The code for reproducibility is available at AF1.

1 Introduction

In recent years, large language models (LLMs) have achieved remarkable progress across multiple domains. For example, LLMs have demonstrated strong reasoning capabilities in structured information understanding and complex question answering tasks (Ye et al., 2026a; Guo et al., 2026; Ye et al., 2026b,c; Guo et al., 2025). Beyond conventional language generation, recent studies have further extended LLMs toward autonomous agents, reinforcement learning-based reasoning, and long-horizon decision-making systems (Hao et al., 2026a,b,c; Wang et al., 2026; Liu et al., 2026). Meanwhile, foundation models have also been increasingly adapted to specialized vision-language applications and domain-specific scenarios (Zhu et al., 2025a, 2026a; Lin et al., 2026; Zhu et al., 2026c, 2024, 2025b, 2026b). However, their performance gains are often closely coupled with the continuous expansion of the model scale. For example, representative open-source models such as LLaMA-3.1-405B (Grattafiori et al., 2024) contain 405B parameters, while the recently released Qwen3.5-397B-A17B (Qwen Team, 2026) reaches 397B total parameters with 17B activated parameters. This rapid scaling leads to substantial memory overhead during inference. For this class of models approaching the 400B-parameter scale, the model weights alone may require memory on the order of 800 GB under BF16/FP16 precision, even before

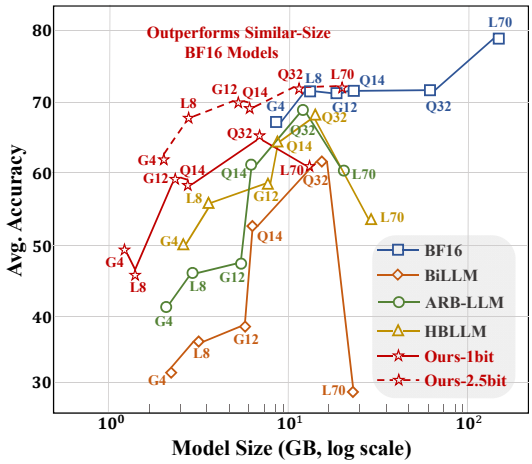


Figure 1: Accuracy–memory trade-off across BF16 models and binarization-based PTQ methods, with L, G, and Q denoting LLaMA-3, Gemma-3, and Qwen3.

* Equal contribution.
† This work was conducted during his internship at Houmo AI.
‡ Corresponding author.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 13527–13558
October 24-29, 2026 ©2026 Association for Computational Linguistics

accounting for additional runtime costs such as the KV cache and intermediate activations. Therefore, reducing the storage and inference costs of LLMs without compromising performance has emerged as a critical challenge for practical deployment, motivating extensive studies on efficient model compression and low-bit quantization.

Post-training quantization (PTQ) has emerged as a key paradigm for reducing the storage and inference costs of LLMs (Zhao et al., 2025b, 2026a; Xu et al., 2026), with recent studies exploring increasingly aggressive low-bit representations beyond conventional quantization (Zhao et al., 2026c,b), among which weight binarization is particularly attractive due to its potential for single-bit storage and bit-level computation. Recent binarization-based PTQ methods, including BiLLM (Huang et al., 2024a), ARB-LLM (Li et al., 2024b), and HBLLM (Chen et al., 2025), mitigate the inherent performance degradation of binarization by leveraging local Hessian information to identify salient weights and allocate additional storage to them. However, as shown in Fig. 1, such designs often come with non-negligible auxiliary metadata, including scaling factors, group-wise parameters, and outlier indices. As a result, the end-to-end bits per weight (BPW) can substantially exceed the nominal 1-bit target, often entering an effective 2–4-bit regime in practice. On the other hand, quantization-aware training (QAT) methods such as OneBit (Xu et al., 2024) can produce models with genuine 1-bit representations, but they typically rely on large-scale training data, long optimization schedules, and substantial computational resources. Therefore, achieving truly 1-bit PTQ without heavy auxiliary metadata or costly retraining remains a critical challenge for end-to-end efficient LLM inference.

To further understand the difficulty of achieving genuine 1-bit quantization under the PTQ setting, we identify two key limitations of existing methods. First, most binarization-based PTQ methods follow a single-matrix binarization paradigm, where a full-precision weight matrix is directly approximated by a scaled binary matrix, i.e., $\mathbf{W} \approx \alpha\mathbf{B}$, with $\mathbf{B} \in \{-1, +1\}^{m\times n}$. However, the limited representational capacity of a single binary matrix makes it difficult to capture the complex magnitude distributions and structural correlations of LLM weights. Unlike QAT, which can refine binary parameters through large-scale data-driven optimization, PTQ must perform discrete reconstruction with limited calibration data and optimization steps, resulting in much stricter error-control requirements. Recent studies on dataset distillation and data-efficient learning have explored reducing data requirements while preserving model performance, providing complementary perspectives for efficient model development (Li et al., 2026c,a,b, 2024a, 2025). Second, different structural units in LLMs exhibit highly heterogeneous sensitivity to binarization errors. Uniform quantization configurations or fixed structural budgets often fail to match the non-uniform importance of layers and submodules, leading to suboptimal capacity allocation and full-model performance degradation.

To this end, we propose **A**ll **f**or **1**-Bit (**AF1**), a genuine 1-bit post-training quantization framework designed to overcome the representational bottleneck of single-matrix binarization and the heterogeneous sensitivity of LLM components. **AF1** operates through two seamlessly coupled mechanisms. First, Null-space-Aware Binary Factorization (NABF) expands 1-bit capacity via a dual-factor binary structure. It constructs a Hessian-aware surrogate space and strategically redirects discrete projection residuals into the approximate null space of the fixed factor, absorbing structural mismatch into lightweight scaling vectors with zero inference overhead. It then performs a low-cost, scale-only global reconstruction to recalibrate continuous scales while keeping binary matrices frozen. Second, Hierarchical Shapley Allocation (HiSA) optimizes the strictly limited structural budget from a global interaction perspective. It first assigns coarse layer-level budgets via progressive Shapley sampling on end-to-end NLL degradation, and then distributes these budgets to intra-layer submodules using dual-sensitivity Shapley estimates based on the directional and magnitude distortions of block outputs. Extensive experiments demonstrate that **AF1** achieves a superior accuracy–memory trade-off over existing binarization-based PTQ methods while maintaining a strict 1.0 effective-BPW storage budget for target linear weights. Our key contributions can be summarized as follows:

- We identify two key obstacles to genuine 1-bit PTQ for LLMs: the limited representational capacity of single-matrix binarization and the heterogeneous sensitivity of layers and submodules under extreme compression.
- We propose **A**ll **f**or **1**-Bit (**AF1**), a genuine 1-bit PTQ framework that combines

NABF for expressive and error-suppressed binary factorization with **HiSA** for hierarchical, interaction-aware structural budget allocation.

- Extensive experiments on LLM families demonstrate that AF1 achieves a superior accuracy–memory trade-off over existing binarization-based PTQ methods while maintaining a strict 1.0 effective-BPW storage budget for target linear weights.

2 Related Work

Binarization for LLMs. Weight binarization represents LLM weights with binary values, e.g., $\{-1, +1\}$, to reduce storage, memory bandwidth, and inference costs. Existing methods mainly compensate for severe reconstruction error. PB-LLM (Shang et al., 2023) preserves salient weights with Hessian-aware compensation, while BiLLM (Huang et al., 2024a) uses weight distribution properties, residual approximation, and group-wise search. ARB-LLM (Li et al., 2024b), STBLLM (Dong et al., 2024), and HBLLM (Chen et al., 2025) further improve fidelity through refined binarization, structured sparsity, frequency-aware grouping, or saliency-driven selection. Recent studies have also explored ternary representations to further extend the boundary of low-bit LLM compression (Yan et al., 2026; Zhao et al., 2026b). Despite their effectiveness, these methods require auxiliary metadata, such as salient weights, fine-grained scales, bitmaps, masks, or grouping statistics, making their effective BPW far above the nominal 1-bit target. Achieving genuine 1.0 effective-BPW linear-weight binarization without excessive reconstruction overhead remains challenging.

Mixed-Precision Quantization for LLMs. Mixed-precision quantization (MPQ) improves the accuracy–efficiency trade-off by assigning different bit-widths to components. APTQ (Guan et al., 2024) combines Hessian trace and attention-aware sensitivity for Transformer-block-level allocation. Recent methods like SpQR (Dettmers et al., 2023) preserve high-precision outliers; Atom (Zhao et al., 2024) combines channel reordering, fine-grained grouping, and mixed precision; SliM-LLM (Huang et al., 2024b) performs salience-based group allocation; CMPQ (Chen et al., 2024) uses activation distributions; and AMQ (Lee et al., 2025) adopts AutoML search for weight-only MPQ. Recent low-bit quantization studies have further investigated robust activation quantization and ternary representations for large-scale models (Yang et al., 2025c; Yan et al., 2026). Related quantization studies beyond LLMs have also explored adaptive search and reconstruction strategies for diffusion transformers and other large-scale models under aggressive compression settings (Yang et al., 2025b; Zhu et al., 2025c; Zhang et al., 2026b). However, these methods largely rely on local sensitivity, saliency, or activation statistics, and rarely model layer–submodule interactions. Since quantization errors can propagate across layers and alter downstream distributions and error accumulation, interaction-aware hierarchical allocation remains important for LLM binarization.

3 Preliminaries

Null Space. Null space describes perturbation directions that are canceled by a linear mapping. Given $\mathbf{B} \in \mathbb{R}^{r\times d}$, a perturbation $\mathbf{E} \in \mathbb{R}^{n\times r}$ satisfying $\mathbf{EB} = \mathbf{0}$ lies in the left null space induced by $\mathbf{B}$. Similarly, given $\mathbf{A} \in \mathbb{R}^{n\times r}$, a perturbation $\mathbf{E} \in \mathbb{R}^{r\times d}$ satisfying $\mathbf{AE} = \mathbf{0}$ lies in the right null space induced by $\mathbf{A}$. Such perturbations do not affect the output of the matrix multiplication, making them useful for compensating discrete projection errors in our binary factorization.

Shapley Value. The Shapley value (Winter, 2002) measures the importance of a participant by averaging its marginal contribution over all possible coalitions. Given participants $\mathcal{N} = \{1, \ldots, n\}$ and a value function $v(S)$ for coalition $S \subseteq \mathcal{N}$, the Shapley value of participant i is

$$\phi_i = \sum_{S\subseteq\mathcal{N}\setminus\{i\}} \frac{|S|!(n-|S|-1)!}{n!} \left[v(S\cup\{i\}) - v(S)\right]. \tag{1}$$

In LLMs, participants can correspond to layers, blocks, or submodules, while $v(S)$ measures the reconstruction change under a given compression state. We use them to allocate structural capacity.

4 Method

Overview. The overview of **AF1** is illustrated in Fig. 2. AF1 integrates **NABF**, which improves binary reconstruction through Hessian-aware surrogate reparameterization, null-space-aware binary ADMM factorization, and scale-only global reconstruction, with **HiSA**, which hierarchically allo-

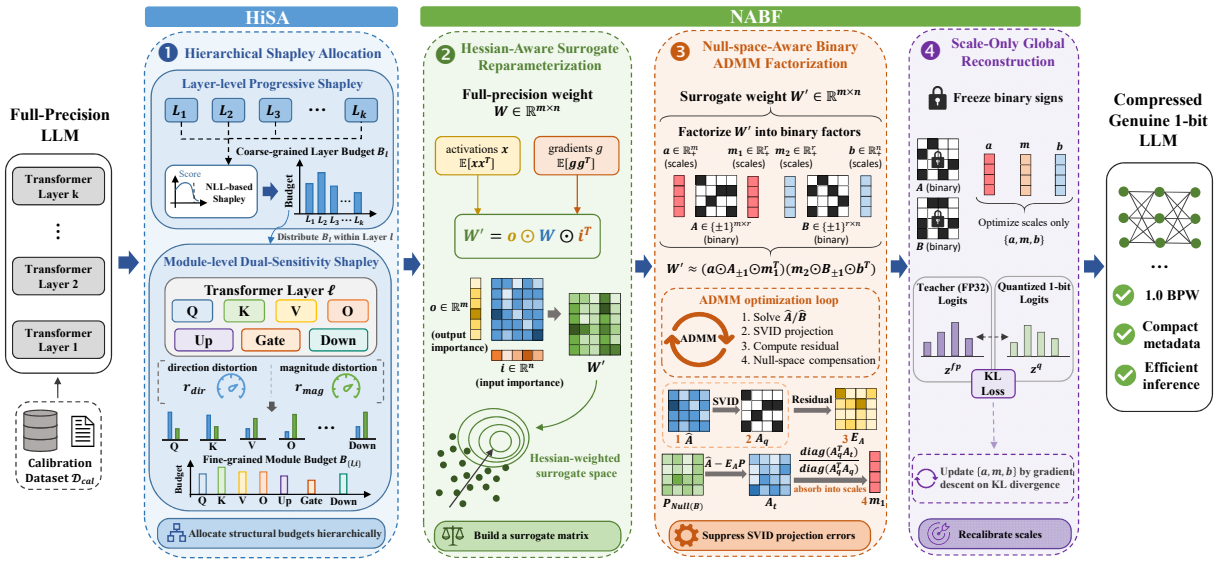


Figure 2: Overview of AF1. Null-space-Aware Binary Factorization (NABF) constructs Hessian-aware surrogate weights, suppresses SVID projection errors, and performs scale-only reconstruction. Hierarchical Shapley Allocation (HiSA) allocates structural budgets from coarse layer-level capacity to fine-grained module-level dimensions.

cates structural budgets according to global layer sensitivity and submodule-level distortion. This design enables accurate LLM compression under a strict 1.0 effective-BPW budget for target linear weights. The detailed algorithmic workflow is provided in Appendix A.1.

4.1 Null-space-Aware Binary Factorization

Hessian-Aware Surrogate Reparameterization. To avoid treating all weight perturbations uniformly under mean-squared reconstruction, we construct a Hessian-aware surrogate weight space. For a linear weight matrix $\mathbf{W}$, a second-order approximation suggests that quantization errors should be weighted by the local Hessian geometry. We follow DBF (Boža and Macko, 2025) and use a K-FAC-inspired diagonal approximation (Martens and Grosse, 2015) based on input activations and output gradients. We define the input importance vector $\mathbf{i}$ and output importance vector $\mathbf{o}$ as

$$i_j = \sqrt{\mathbb{E}[x_j^2]}, \qquad o_u = \sqrt{\mathbb{E}[g_u^2]}, \tag{2}$$

where $\mathbf{x}$ denotes the input activation and $\mathbf{g} = \partial\mathcal{L}/\partial\mathbf{Y}$ denotes the output gradient. The original weight matrix is then reparameterized as

$$\mathbf{W}' = \mathbf{o} \odot \mathbf{W} \odot \mathbf{i}^\top. \tag{3}$$

This converts the Hessian-weighted reconstruction into a Frobenius objective in the surrogate space, so that subsequent binary factorization emphasizes task-sensitive input and output channels.

Null-space-Aware Binary ADMM Factorization. Given the Hessian-aware surrogate matrix $\mathbf{W}'$, we first follow DBF (Boža and Macko, 2025) and decompose it into two binary sign factors with continuous scaling vectors to enhance representation capacity (detailed proof provided in Appendix A.2.1):

$$\mathbf{W}' \approx \left(\mathbf{a} \odot \mathbf{A}_{\pm1} \odot \mathbf{m}_1^\top\right)\left(\mathbf{m}_2 \odot \mathbf{B}_{\pm1} \odot \mathbf{b}^\top\right), \tag{4}$$

where $\mathbf{A}_{\pm1}$ and $\mathbf{B}_{\pm1}$ are binary sign matrices, and $\{\mathbf{a}, \mathbf{m}_1, \mathbf{m}_2, \mathbf{b}\}$ are full-precision scaling vectors. We optimize this factorization using the framework of DBF (Boža and Macko, 2025): fixing one factor, ADMM solves a continuous proxy, which SVID projects back to the binary-scaled domain (Appendix A.1.4). Taking the left-factor update as an example, let $\hat{\mathbf{A}}$ denote the continuous proxy and $\mathbf{A}_q = \mathrm{SVID}(\hat{\mathbf{A}})$ denote its discrete projection. The residual introduced by discrete projection is

$$\mathbf{E}_A = \hat{\mathbf{A}} - \mathbf{A}_q. \tag{5}$$

This residual propagates through the fixed factor and causes non-negligible output perturbations. The core innovation of NABF is to suppress this propagated projection error through null-space-aware compensation. Taking the left-factor update as an example, the actual perturbation is not $\mathbf{E}_A$ itself but $\mathbf{E}_A\mathbf{B}$. If $\mathbf{E}_A$ lies in the approximate left null space of the fixed factor $\mathbf{B}$, then $\mathbf{E}_A\mathbf{B} \approx \mathbf{0}$, so the projection error can be largely canceled in the forward mapping (Fang et al., 2024). Therefore, instead of only minimizing the residual norm,

we guide its compensable component toward the null-space directions of the fixed factor.

To construct this subspace efficiently, we avoid directly computing the null space of $\mathbf{B}$. Following existing null-space projection strategies (Wang et al., 2021), we use the uncentered covariance matrix $\mathbf{C}_B = \mathbf{B}\mathbf{B}^\top$, whose null eigenspace is equivalent to the left null space of $\mathbf{B}$ (see Appendix A.2.2). Then, we perform SVD on $\mathbf{C}_B$ as $\mathbf{C}_B = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ and collect the trailing singular vectors selected by an adaptive residual-to-total energy cutoff (Jolliffe and Cadima, 2016) into $\widehat{\mathbf{U}}$, with the threshold set to $\eta = 0.01$ (see Appendix A.2.3). The approximate null-space projector is defined as $\mathbf{P}_B = \widehat{\mathbf{U}}\widehat{\mathbf{U}}^\top$, satisfying $\mathbf{P}_B\mathbf{B} \approx \mathbf{0}$ (see Appendix A.2.4).

This property implies that residual components projected onto this subspace have negligible effect after multiplication by the fixed factor $\mathbf{B}$. We therefore define the null-space-compensated target as

$$\mathbf{A}_{\text{target}} = \hat{\mathbf{A}} - \mathbf{E}_A\mathbf{P}_B. \tag{6}$$

Multiplying both sides by $\mathbf{B}$ gives

$$\begin{aligned}\mathbf{A}_{\text{target}}\mathbf{B} &= \left(\hat{\mathbf{A}} - \mathbf{E}_A\mathbf{P}_B\right)\mathbf{B} \\ &= \hat{\mathbf{A}}\mathbf{B} - \mathbf{E}_A\mathbf{P}_B\mathbf{B} \approx \hat{\mathbf{A}}\mathbf{B}.\end{aligned} \tag{7}$$

Thus, $\mathbf{A}_{\text{target}}$ preserves the propagated behavior of the continuous proxy while providing a better target for recalibrating the discrete factor. To incorporate this compensated target back into the double-factor binary parameterization in Eq. 4, we absorb the compensation through the existing scaling degrees of freedom. Specifically, for the left-factor update, we fit a column-wise correction vector $\boldsymbol{\gamma}$ along the intermediate dimension:

$$\boldsymbol{\gamma}^* = \arg\min_{\boldsymbol{\gamma}} \|\mathbf{A}_{\text{target}} - \mathbf{A}_q \operatorname{Diag}(\boldsymbol{\gamma})\|_F^2. \tag{8}$$

This convex least-squares problem has the closed-form solution (the proof given in Appendix A.2.5):

$$\boldsymbol{\gamma}^* = \frac{\operatorname{diag}\left(\mathbf{A}_q^\top \mathbf{A}_{\text{target}}\right)}{\operatorname{diag}\left(\mathbf{A}_q^\top \mathbf{A}_q\right)}. \tag{9}$$

We fold the correction into the intermediate scaling vector, i.e., $\mathbf{m}_1 \leftarrow \mathbf{m}_1 \odot \boldsymbol{\gamma}^*$. This update changes neither the binary sign matrices nor the inference-time computation graph (right-factor update is symmetric), enabling NABF to absorb discrete projection errors through scale recalibration while preserving the 1-bit parameterization.

Scale-Only Global Reconstruction. After double-factor binary optimization, we freeze all binary sign matrices and perform the final correction only through the continuous scale parameters. For the l-th linear layer, the reconstructed weight is

$$\hat{\mathbf{W}}^{(l)} = \left(\mathbf{a}^{(l)} \odot \mathbf{A}_{\pm 1}^{(l)} \odot (\mathbf{m}^{(l)})^\top\right)\left(\mathbf{B}_{\pm 1}^{(l)} \odot (\mathbf{b}^{(l)})^\top\right), \tag{10}$$

where $\mathbf{A}_{\pm 1}^{(l)}$ and $\mathbf{B}_{\pm 1}^{(l)}$ are fixed binary sign matrices, $\mathbf{m}^{(l)} = \mathbf{m}_1^{(l)} \odot \mathbf{m}_2^{(l)}$ denotes the merged intermediate scale from the two factorization scales, and $\mathbf{a}^{(l)}$, $\mathbf{m}^{(l)}$, and $\mathbf{b}^{(l)}$ are learnable scales. We collect learnable scales as $\mathcal{S} = \{\mathbf{a}^{(l)}, \mathbf{m}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{L}$. Given a small calibration set $\mathcal{D}_{\text{cal}}$, we use the full-precision model M as a teacher and optimize $\mathcal{S}$ so that the compressed model $\hat{M}(\cdot;\mathcal{S})$ matches the teacher distribution (Kwon et al., 2022). Let $\mathbf{z}_T(\mathbf{X})$ and $\mathbf{z}_Q(\mathbf{X};\mathcal{S})$ denote the teacher and compressed-model logits. The scale-only objective is

$$\begin{aligned}\min_{\mathcal{S}} \quad \mathbb{E}_{\mathbf{X}\sim\mathcal{D}_{\text{cal}}}\Big[D_{\text{KL}}\big(\operatorname{softmax}(\mathbf{z}_T(\mathbf{X})) \,\|\, \\ \operatorname{softmax}(\mathbf{z}_Q(\mathbf{X};\mathcal{S}))\big)\Big].\end{aligned} \tag{11}$$

This stage recalibrates only the scale space while preserving the double-factor binary computation graph. Since binary matrices are frozen, the trainable parameters per linear layer are reduced from $O(d_{\text{out}}r + rd_{\text{in}})$ to $O(d_{\text{out}} + r + d_{\text{in}})$. Thus, NABF bridges performance gaps with minimal memory while preserving the 1-bit parameterization.

4.2 Hierarchical Shapley Allocation

Although NABF achieves accurate operator-level reconstruction under a given BPW budget, applying a uniform BPW configuration to the full model ignores the heterogeneous sensitivity of different layers and submodules. However, conventional local sensitivity estimates can be unreliable because LLM compression errors propagate across layers and interact with each other (Zhao et al., 2025a). We therefore propose Hierarchical Shapley Allocation (HiSA), which evaluates component importance from a cooperative-game perspective. By capturing global interactions and within-layer heterogeneity under fixed storage constraints, HiSA generates adaptive BPW configurations for NABF.

Layer-Level Coarse-Grained Budget Allocation via Progressive Shapley. We formulate layer-wise dimension assignment as a structural-capacity game, whose interaction-aware property is discussed in Appendix A.2.6. Following Shapley-based mixed-precision quantization (Zhao et al.,

Method	#Bits (W)	LLaMA-3-8B		LLaMA-3-70B		Qwen3-8B		Qwen3-14B		Qwen3-32B		Gemma-3-4B-it		Gemma-3-12B-it	
		0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)
BF16	16	72.51	6.14	79.08	2.86	69.09	9.00	72.47	8.64	72.42	7.61	67.12	17.39	72.20	25.20
RTN	1.00	25.82	2.57e6	24.88	3.77e4	25.28	7.81e6	25.55	1.37e8	25.41	3.82e6	25.28	2.12e7	25.30	7.29e7
BiLLM	2.88	35.38	48.18	28.32	137.64	33.70	96.21	53.53	28.66	61.23	16.79	32.33	134.42	38.41	381.19
STBLLM$_{(4:8)}$	3.50	29.43	196.03	26.55	963.92	28.02	291.76	40.88	48.80	55.86	17.31	29.31	202.54	31.46	4.69e5
ARB-LLM$_{RC}$	2.51	46.21	28.97	60.76	11.10	49.85	40.90	61.60	18.40	<u>69.56</u>	12.93	41.83	95.18	47.15	4.68e3
HBLLM$_{row}$	3.25	<u>55.35</u>	<u>14.26</u>	53.19	9.55	54.76	16.28	<u>64.12</u>	<u>11.01</u>	67.28	<u>10.27</u>	<u>50.35</u>	43.05	57.40	758.45
AF1	2.50	**67.35**	**8.07**	**72.27**	**4.52**	**67.95**	**9.71**	**70.33**	**8.87**	**72.49**	**8.38**	**62.29**	**15.39**	**70.44**	**15.88**
AF1	**1.00**	45.48	22.18	<u>61.80</u>	<u>8.91</u>	<u>54.82</u>	<u>16.26</u>	59.61	13.32	64.26	11.28	49.86	<u>32.82</u>	<u>59.02</u>	<u>27.07</u>

Table 1: Comparison of perplexity on WikiText2 and averaged accuracy on seven zero-shot tasks. Bold and underline denote the **best** and <u>second-best</u> quantized results. Full results are in Appendix A.4.1.

2025a), we evaluate the structural importance of each layer by changing its intermediate dimension. Let $T = \{1, 2, \ldots, L\}$ denote the layers. For any subset $S \subseteq T$, layers in S keep the original intermediate dimension r_{orig}, while the remaining layers are degraded to a proxy dimension r_{proxy} that satisfies the 1.0 effective-BPW budget for target linear weights. Given a calibration corpus $\mathcal{D}$, we evaluate each mixed-capacity state by the average token-level negative log-likelihood:

$$v_{\text{NLL}}(S) = \mathbb{E}_{(x,t)\sim\mathcal{D}}\left[-\log p(x_{t+1} \mid x_{\leq t}; S)\right], \quad (12)$$

where a larger value indicates worse performance. Since enumerating all states is infeasible, we estimate Shapley scores by Monte Carlo permutations. For a sampled permutation $\pi = (\pi_1, \ldots, \pi_L)$, we start from the full-capacity state and progressively remove layers from S, i.e., degrade them to r_{proxy}. Let S_ℓ be the set of layers still retaining r_{orig} before degrading π_ℓ. The marginal contribution of π_ℓ is the induced NLL increase:

$$\Delta v_{\pi_\ell} = v_{\text{NLL}}(S_\ell \setminus \{\pi_\ell\}) - v_{\text{NLL}}(S_\ell). \quad (13)$$

Averaging over M sampled permutations gives

$$\Phi_l = \frac{1}{M}\sum_{m=1}^{M} \Delta v_l^{(m)}. \quad (14)$$

We convert the layer-level Shapley scores into coarse structural budgets through a temperature-controlled softmax:

$$B_l = B_{\text{total}} \frac{\exp(\Phi_l/\tau)}{\sum_{j=1}^{L}\exp(\Phi_j/\tau)}. \quad (15)$$

Here, B_{total} is the total structural budget, and τ controls the allocation concentration. The resulting B_l serves as the layer-level budget for subsequent within-layer module allocation.

Block-Level Fine-Grained Allocation via Dual-Sensitivity Shapley. To obtain finer-grained allocation within each layer, HiSA evaluates submodule importance by the output perturbation caused by compressing each module. However, the conventional MSE-based perturbation measure is limited, as it merges directional and magnitude drift into a single scalar despite their different failure modes. HiSA therefore measures two complementary distortions and estimates their Shapley marginal contributions separately. Specifically, r_{dir} captures angular distortion, while r_{mag} captures relative magnitude perturbation (with their formal definitions and distinct failure-mode analysis provided in Appendix A.2.7). We then construct a module-level cooperative game inside layer l. Let H_l^{ref} be the original reference output of layer l, $\mathcal{N}_l$ be the candidate submodules, and $H_l(S)$ be the layer output when a subset $S \subseteq \mathcal{N}_l$ is degraded. For each distortion type $c \in \{\text{dir}, \text{mag}\}$, we define the subset error as

$$E_c(S) = r_c\Big(H_l^{\text{ref}}, H_l(S)\Big). \quad (16)$$

Given a Monte Carlo permutation π, we progressively degrade submodules from $S_0 = \varnothing$. To reduce negative marginals caused by accidental error cancellation, we use the monotonic accumulated error $\hat{E}_c(S_t) = \max(E_c(S_t), \hat{E}_c(S_{t-1}))$ and define the marginal contribution of module $i = \pi_t$ as

$$\Delta_c^{(i)}(\pi) = \hat{E}_c(S_t) - \hat{E}_c(S_{t-1}). \quad (17)$$

Following the same Monte Carlo averaging form as Eq. 14, we obtain the module-level Shapley estimates $\Phi_c^{(i)}$ for $c \in \{\text{dir}, \text{mag}\}$. Because the two scores have different scales, we normalize them with a MAD-based robust Z-score (Iglewicz and Hoaglin, 1993), equivalently written in the scaled-MAD form:

$$z_c^{(i)} = \frac{\Phi_c^{(i)} - \text{Median}(\Phi_c)}{1.4826 \cdot \text{MAD}(\Phi_c) + \epsilon}, c \in \{\text{dir}, \text{mag}\}. \quad (18)$$

Model	Method	#Bits(W)	Model Size	MMLU-Pro	HumanEval	GSM8K
LLaMA-3.1 -8B-Instruct	BF16	16	16.07GB	46.64	67.07	83.24
	BiLLM	2.88	3.75GB	8.77	15.33	21.36
	STBLLM$_{(4:8)}$	3.50	4.32GB	13.74	10.15	10.83
	ARB-LLM$_{RC}$	2.51	3.36GB	10.16	20.12	30.38
	HBLLM$_{row}$	3.25	4.15GB	**23.57**	28.97	40.26
	AF1	**1.00**	**1.56GB**	20.12	**36.71**	**44.78**
Qwen3-14B -Instruct	BF16	16	29.51GB	69.11	89.63	94.77
	BiLLM	2.88	6.70GB	23.75	19.74	21.06
	STBLLM$_{(4:8)}$	3.50	8.11GB	13.38	13.70	12.05
	ARB-LLM$_{RC}$	2.51	5.99GB	26.64	26.37	37.74
	HBLLM$_{row}$	3.25	7.43GB	**43.84**	**52.32**	**72.48**
	AF1	**1.00**	**2.92GB**	38.57	49.76	64.66

Table 2: Results on instruction-tuned models across three challenging benchmarks. Bold and underline denote the **best** and second-best quantized results.

Then, we map the normalized scores by $P_c^{(i)} = 1/(1+\exp(-z_c^{(i)}))$ (Zhang et al., 2026a). To prevent vulnerability in either failure mode from being suppressed by averaging, we fuse the two modes with a Soft-OR rule $v_{\text{base}}^{(i)} = 1-(1-P_{\text{dir}}^{(i)})(1-P_{\text{mag}}^{(i)})$. Since submodules within the same layer can have substantially different parameter counts, we allocate the layer budget using a smoothed size-aware sensitivity (see Appendix A.2.8):

$$b_i = B_l \cdot \frac{\left(\sum_{k\in\mathcal{N}_l}|\mathbf{W}_k|\right)\left(v_{\text{base}}^{(i)}/|\mathbf{W}_i|+\epsilon\right)^{\alpha}}{\sum_{j\in\mathcal{N}_l}|\mathbf{W}_j|\left(v_{\text{base}}^{(j)}/|\mathbf{W}_j|+\epsilon\right)^{\alpha}}. \quad (19)$$

This allocation preserves the layer-level budget and the resulting b_i is then converted into the intermediate dimension used by NABF.

5 Experiments

5.1 Experiment setup

Models and Datasets. We evaluate **AF1** on representative LLM families, including LLaMA-2/3 (Grattafiori et al., 2024), Qwen3 (Yang et al., 2025a), and Gemma-3 (Team et al., 2025), along with their instruction-tuned variants. We report WikiText2 perplexity (Merity et al., 2016) and seven zero-shot tasks: ARC-Challenge/Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), LAMBADA-openai/standard (Paperno et al., 2016), PIQA (Bisk et al., 2019), and WinoGrande (Sakaguchi et al., 2019). We also include stronger reasoning and generation benchmarks: MMLU-Pro (Wang et al., 2024), GSM8K (Cobbe et al., 2021), and HumanEval (Chen et al., 2021).

Baseline Methods. We compare **AF1** with representative binarization-based PTQ methods, including BiLLM (Huang et al., 2024a), ARB-LLM (Li et al., 2024b), STBLLM (Dong et al., 2024), and HBLLM (Chen et al., 2025) (with their effective BPW derived in Appendix A.3). To provide additional reference under genuine 1-bit compression, we also include recent QAT methods, including OneBit (Xu et al., 2024) and LittleBit (Lee et al., 2026). Since QAT requires extra data and much higher optimization cost, these results serve as reference points rather than strict PTQ comparisons.

Implementation Details. All experiments are implemented with PyTorch and conducted on NVIDIA RTX PRO 6000 GPUs. Following the calibration setting of DBF, we randomly sample 128 sequences from the RedPajama (Weber et al., 2024) dataset with a sequence length of 2048. For the ADMM-SVID optimization in NABF, we follow the default setting of DBF. For evaluation, we use lm-evaluation-harness (Gao et al., 2024) for the seven zero-shot tasks, and EvalScope (Team, 2024) for MMLU-Pro, GSM8K, and HumanEval.

5.2 Main Results

Comparison Results. Table 1 reports the main results of AF1 across LLaMA-3, Qwen3, and Gemma-3 model families. Under the 2.5-bit setting, which is close to the BPW of existing binarization-based PTQ methods, AF1 achieves the best results among quantized methods on all models. Compared with the strongest baseline HBLLM, AF1 improves the average zero-shot accuracy across all models by over 20% and reduces the average WikiText2 PPL by over 90%, showing substantially stronger low-bit reconstruction capability under comparable storage cost. More importantly, AF1 remains highly competitive under a genuine 1-bit storage budget. On the largest model, LLaMA-3-70B, AF1 achieves 61.80 zero-shot accuracy and 8.91 WikiText2 PPL at 1.0 effective BPW for target linear weights, even surpassing the strongest baseline HBLLM. Similar trends on Qwen3 and Gemma-3 further indicate that AF1 delivers stable cross-family 1-bit reconstruction rather than overfitting to a single architecture (Practical generation examples with GPT-5.5 (OpenAI, 2026) annotations are provided in Appendix A.5).

Experiments on Instruction-tuned Models. Instruction-tuned models are practically important but more vulnerable to quantization errors in reasoning, coding, and mathematics. As shown in Table 2, AF1 maintains strong instruction-tuned performance with much smaller model sizes. For example, on Qwen3-14B-Instruct, AF1 consistently

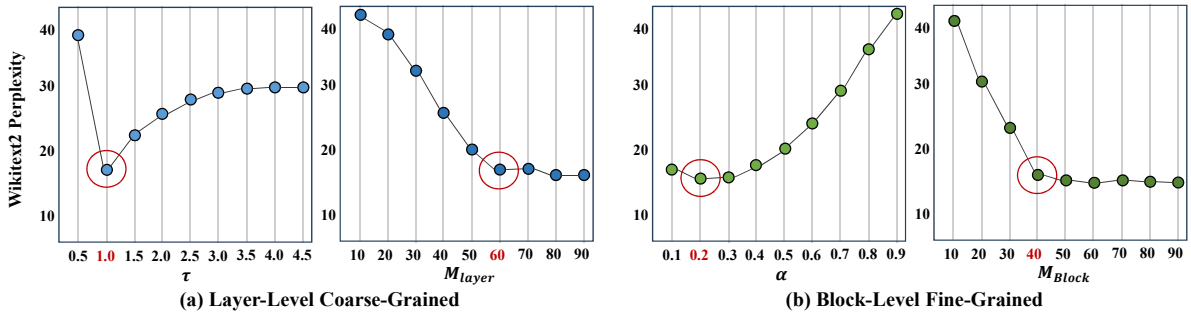


Figure 3: Hyperparameter sensitivity of HiSA on WikiText2 perplexity. (a) Layer-level coarse-grained allocation under different τ and M_{layer}. (b) Block-level fine-grained allocation under different α and M_{block}.

Model	NABF Step1	NABF Step2	NABF Step3	HiSA Layer	HiSA Module	Wiki2 PPL(↓)	0-shot[7] Avg.(↑)
LLaMA-3.2-3B	✗	✗	✗	✗		8.9e2	25.48
	✓	✗	✗			1.3e2	29.31
	✓	✓	✗			41.34	34.85
	✓	✓	✓			30.47	40.79
	✓			✓	✗	25.52	42.30
				✓	✓	**24.21**	**43.09**

Table 3: Combined ablation on LLaMA-3.2-3B. Step1–3 denote surrogate reparameterization, null-space binary factorization, and scale-only reconstruction; Layer and Module denote the two stages of HiSA allocation.

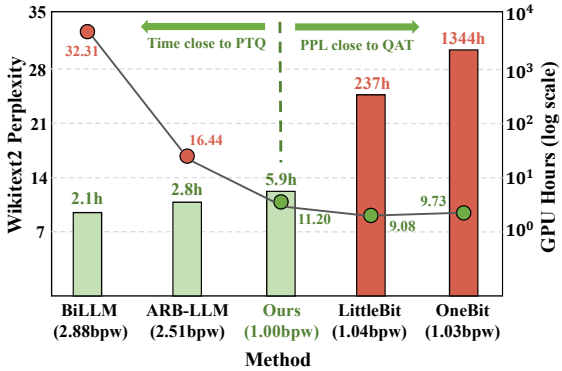


Figure 4: Quantization time and WikiText2 perplexity comparison under near-1-bit compression.

outperforms BiLLM, STBLLM, and ARB-LLM across all benchmarks. Compared with HBLLM, AF1 has only about a 9% lower average score while using just 39.3% of its model size (7.43 GB → 2.92 GB). These results show AF1 achieves a favorable accuracy–memory trade-off for instruction-tuned LLMs under genuine 1-bit PTQ.

5.3 Ablation Studies

Component Ablation. We analyze the contributions of NABF and HiSA using WikiText2 PPL and seven-task average accuracy. As shown in Table 3, NABF alone substantially improves the 1-bit reconstruction quality, reducing PPL from $8.9e2$ to 30.47 and achieving 40.79 average accuracy. Building on this stronger reconstruction, HiSA further improves the limited-budget allocation: the final model reduces PPL to 24.21 and raises the average accuracy to 43.09. Additional ablations on the residual-energy threshold and HiSA sensitivity metrics are provided in Appendices A.4.2 and A.4.3. These results show that NABF and HiSA are complementary: NABF provides a high-fidelity binary reconstruction, while HiSA allocates the scarce capacity to sensitive layers and submodules.

Hyperparameter Sensitivity of HiSA. We analyze HiSA's sensitivity on Qwen3-8B with respect to four key hyperparameters: the temperature coefficient τ in Eq. 15, the smoothing exponent α in Eq. 19, and the layer- and block-level sampling numbers M_{layer} and M_{block}. As shown in Fig. 3, τ controls the concentration of layer-wise budgets: overly small values over-concentrate capacity on a few layers, while overly large values approach uniform allocation. The best result is obtained at $\tau = 1.0$. Increasing M_{layer} and M_{block} stabilizes Shapley estimation, saturating around 60 and 40 samples, respectively. α balances sensitivity-density redistribution and noise amplification, with the best result at $\alpha = 0.2$. Similar trends on other models are reported in Appendix A.4.4, which supports our default configuration of $\tau = 1.0$, $M_{\text{layer}}/M_{\text{block}} = 60/40$, and $\alpha = 0.2$. We also report the calibration ablation in Appendix A.4.5.

5.4 Efficiency Analysis of AF1

Quantization Time Comparison. We compare AF1 with representative PTQ and QAT-based binarization methods on LLaMA-2-7B. As shown in Fig. 4, prior PTQ methods such as BiLLM and ARB-LLM require only a few GPU hours, but their BPWs are close to 3 and PPL are much worse. In contrast, QAT-based methods such as LittleBit and

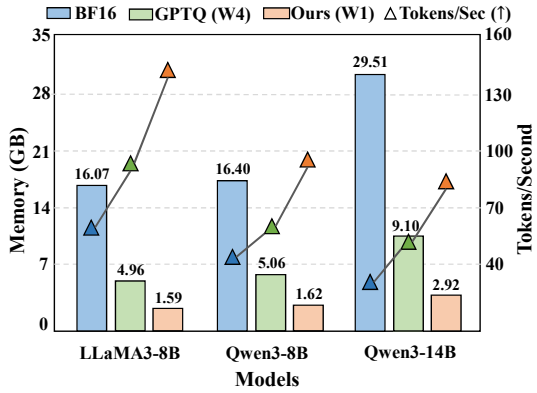


Figure 5: Inference memory and decoding throughput comparison across BF16, GPTQ-W4, and AF1.

OneBit achieve lower PPL near 1-bit storage, but require hundreds to thousands of GPU hours, limiting scalability. As a PTQ method, AF1 bridges this gap: under a 1.0 effective-BPW target-linear-weight setting, it reduces PPL from 16.44 (ARB-LLM) to 11.20 with only 5.9 GPU hours. Compared with QAT baselines, AF1 incurs only a small PPL gap relative to them while being about 40× faster than LittleBit and 228× faster than OneBit. These results show that AF1 provides PTQ-level efficiency and near-QAT performance.

Speedup and Memory Savings. We benchmark the inference efficiency of AF1 on three models, comparing BF16, GPTQ-W4, and AF1 in memory usage and decoding throughput, with binary matrix multiplication implemented using GemLite (Badri and Shaji, 2023). Under a batch size of 1, a 1024-token prefill, and 4096-token decoding, Fig. 5 shows that AF1 consistently reduces memory and improves throughput. Since long-context autoregressive decoding is memory-bandwidth-bound, reducing BPW to 1 substantially lowers memory traffic compared with BF16 and 4-bit PTQ, leading to faster decoding. Across three models, AF1 reduces memory by about 90% over BF16 and 68% over GPTQ-W4 on average, while improving decoding throughput by about 2.5× over BF16. These results highlight the practical memory and decoding efficiency gains enabled by AF1 in practice.

6 Conclusion

We present AF1, a genuine 1-bit PTQ framework for LLMs. By addressing the key challenges of low-capacity binary reconstruction, projection-error propagation, and non-uniform structural sensitivity, AF1 effectively enables binarization without costly QAT. Specifically, NABF enhances binary reconstruction, while HiSA allocates the limited structural budget across layers and submodules. Experiments show that AF1 substantially outperforms existing binary PTQ methods under comparable or lower BPW, approaches the accuracy of QAT-based binarization methods with lower quantization cost, and brings practical memory reduction and decoding acceleration. These results demonstrate that AF1 offers a scalable and deployment-friendly pathway toward genuine 1-bit LLM compression.

Limitations

AF1 focuses on weight-only post-training binarization under a strict 1.0 effective-BPW budget for target linear weights. This setting substantially reduces model memory and decoding traffic, but it does not directly compress activations or the KV cache, which can become important in long-context or large-batch serving. In addition, our efficiency evaluation is based on a representative GPU implementation with optimized binary matrix multiplication; further system-level integration into broader inference stacks remains an important future direction. Finally, our experiments cover representative LLaMA, Qwen, and Gemma models and standard language benchmarks, while evaluations on more architectures, broader instruction-tuned variants, and real deployment workloads would further validate generality.

Ethical Considerations

AF1 is a model compression method and does not introduce new training data or modify the semantic objectives of the original models. Its main societal benefit is reducing the memory and bandwidth cost of LLM deployment, which can improve accessibility and lower energy consumption. However, more efficient LLM deployment may also lower the barrier to misuse of existing models, such as automated spam, misinformation generation, or other harmful applications. We therefore encourage deploying AF1-compressed models with the same safety, monitoring, and access-control mechanisms used for the corresponding full-precision models.

References

Hicham Badri and Appu Shaji. 2023. Half-quadratic quantization of large machine learning models.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Vladimír Boža and Vladimír Macko. 2025. Addition is almost all you need: Compressing neural networks with double binary factorization. *arXiv preprint arXiv:2505.11076*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Ningning Chen, Weicai Ye, and Ying Jiang. 2025. Hbllm: A haar-based approach for accurate structured 1-bit quantized llms. *arXiv e-prints*, pages arXiv–2512.

Zihan Chen, Bike Xie, Jundong Li, and Cong Shen. 2024. Channel-wise mixed-precision quantization for large language models. *arXiv preprint arXiv:2410.13056*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

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

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*.

Peijie Dong, Lujun Li, Yuedong Zhong, Dayou Du, Ruibo Fan, Yuhan Chen, Zhenheng Tang, Qiang Wang, Wei Xue, Yike Guo, and 1 others. 2024. Stbllm: Breaking the 1-bit barrier with structured binary llms. *arXiv preprint arXiv:2408.01803*.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2024. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Ziyi Guan, Hantao Huang, Yupeng Su, Hong Huang, Ngai Wong, and Hao Yu. 2024. Aptq: Attention-aware post-training mixed-precision quantization for

large language models. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6.

Yu Guo, Dong Jin, Shenghao Ye, Shuangwu Chen, Jian Yang, and Xiaobin Tan. 2025. SQLForge: Synthesizing reliable and diverse data to enhance text-to-SQL reasoning in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8441–8452, Vienna, Austria. Association for Computational Linguistics.

Yu Guo, Shenghao Ye, Shuangwu Chen, Zijian Wen, Tao Zhang, Bai Qirui, Dong Jin, Yunpeng Hou, Huasen He, Jianyang, and Xiaobin Tan. 2026. Rethinking table pruning in TableQA: From sequential revisions to gold trajectory-supervised parallel search. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12960–12976, San Diego, California, United States. Association for Computational Linguistics.

Zhezheng Hao, Hong Wang, Haoyang Liu, Jian Luo, Jiarui Yu, Hande Dong, Qiang Lin, Can Wang, and Jiawei Chen. 2026a. Rethinking entropy interventions in rlvr: An entropy change perspective. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31105–31133.

Zhezheng Hao, Hong Wang, Jian Luo, Jianqing Zhang, Yuyan Zhou, Qiang Lin, Can Wang, Hande Dong, and Jiawei Chen. 2026b. Recreate: Reasoning and creating domain agents driven by experience. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31018–31046.

Zhezheng Hao, Tianfu Wang, Huanshuo Dong, Ziyan Liu, Hong Wang, Xiankun Lin, Qiang Lin, Can Wang, Hande Dong, and Jiawei Chen. 2026c. Evolve as a team: Collaborative self-evolution for llm-based multi-agent systems. *arXiv preprint arXiv:2605.29790*.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. 2024a. Billm: Pushing the limit of post-training quantization for llms. *Preprint*, arXiv:2402.04291.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qinshuo Liu, Xianglong Liu, Luca Benini, Michele Magno, Shiming Zhang, and Xiaojuan Qi. 2024b. Slim-llm: Salience-driven mixed-precision quantization for large language models. *arXiv preprint arXiv:2405.14917*.

Boris Iglewicz and David C Hoaglin. 1993. *Volume 16: how to detect and handle outliers*. Quality Press.

Ian T. Jolliffe and Jorge Cadima. 2016. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374.

Se Jung Kwon, Jeonghoon Kim, Jeongin Bae, Kang Min Yoo, Jin-Hwa Kim, Baeseong Park, Byeongwook Kim, Jung-Woo Ha, Nako Sung, and Dongsoo Lee. 2022. Alphatuning: Quantization-aware parameter-efficient adaptation of large-scale pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3288–3305.

Banseok Lee, Dongkyu Kim, Youngcheon You, and Youngmin Kim. 2026. Littlebit: Ultra low-bit quantization via latent factorization. *Advances in Neural Information Processing Systems*, 38:116379–116411.

Sangjun Lee, Seung-taek Woo, Jun-gyu Jin, Changhun Lee, and Eunhyeok Park. 2025. Amq: Enabling automl for mixed-precision weight-only quantization of large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35520–35538.

Muquan Li, Hang Gou, Yingyi Ma, Rongzheng Wang, Ke Qin, and Tao He. 2026a. Fixed anchors are not enough: Dynamic retrieval and persistent homology for dataset distillation. *Preprint*, arXiv:2602.24144.

Muquan Li, Hang Gou, Dongyang Zhang, Shuang Liang, Xiurui Xie, Deqiang Ouyang, and Ke Qin. 2026b. Beyond random: Automatic inner-loop optimization in dataset distillation. *Preprint*, arXiv:2510.04838.

Muquan Li, Yingyi Ma, Yihong Huang, Hang Gou, Ke Qin, Ming Li, Yuan-Fang Li, and Tao He. 2026c. Mind your margin and boundary: Are your distilled datasets truly robust? *Preprint*, arXiv:2605.20606.

Muquan Li, Dongyang Zhang, Qiang Dong, Xiurui Xie, and Ke Qin. 2025. Adaptive dataset quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 12093–12101.

Muquan Li, Dongyang Zhang, Tao He, Xiurui Xie, Yuan-Fang Li, and Ke Qin. 2024a. Towards effective data-free knowledge distillation via diverse diffusion augmentation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 4416–4425. ACM.

Zhiteng Li, Xianglong Yan, Tianao Zhang, Haotong Qin, Dong Xie, Jiang Tian, zhongchao shi, Linghe Kong, Yulun Zhang, and Xiaokang Yang. 2024b. Arb-llm: Alternating refined binarizations for large language models. *Preprint*, arXiv:2410.03129.

Jianxin Lin, Chunzheng Zhu, Peter J Kneuertz, Yunfei Bai, and Yuan Xue. 2026. Medcausalx: Adaptive causal reasoning with self-reflection for trustworthy medical vision-language models. *arXiv preprint arXiv:2603.23085*.

Ziyan Liu, Zhezheng Hao, Yeqiu Chen, Hong Wang, Jingren Hou, Ruiyi Ding, Yongkang Yang, Wence Ji, Wei Xia, and Feng Liu. 2026. Meta-cognitive memory policy optimization for long-horizon llm agents. *arXiv preprint arXiv:2605.30159*.

James Martens and Roger Grosse. 2015. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Parikh Neal, Chu Eric, Peleato Borja, and Eckstein Jonathan. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122.

OpenAI. 2026. GPT-5.5 System Card. https://openai.com/index/gpt-5-5-system-card/. Accessed: 2026-05-14.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *Preprint*, arXiv:1606.06031.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *Preprint*, arXiv:1907.10641.

Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. 2023. Pb-llm: Partially binarized large language models. *Preprint*, arXiv:2310.00034.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

ModelScope Team. 2024. EvalScope: Evaluation framework for large models.

Hong Wang, Zhezheng Hao, Jian Luo, Chenxing Wei, Yao Shu, Lei Liu, Qiang Lin, Hande Dong, and Jiawei Chen. 2026. Scheduling your llm reinforcement learning with reasoning trees. In *International Conference on Learning Representations*, volume 2026, pages 154734–154753.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. 2021. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 184–193.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Maurice Weber, Daniel Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. 2024. Redpajama: an open dataset for training large language models. *Preprint*, arXiv:2411.12372.

Eyal Winter. 2002. The shapley value. *Handbook of game theory with economic applications*, 3:2025–2054.

Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. 2024. Onebit: Towards extremely low-bit large language models. *Advances in Neural Information Processing Systems*, 37:66357–66382.

Zukang Xu, Zhixiong Zhao, Xing Hu, Zhixuan Chen, and Dawei Yang. 2026. Kbvq-moe: Klt-guided svd with bias-corrected vector quantization for moe large language models. *arXiv preprint arXiv:2602.11184*.

Xianglong Yan, Chengzhu Bao, Zhiteng Li, Tianao Zhang, Kaicheng Yang, Haotong Qin, Ruobing Xie, Samm Sun, and Yulun Zhang. 2026. Pt ^{2} -llm: Post-training ternarization for large language models. In *International Conference on Learning Representations*, volume 2026, pages 11677–11689.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Kaicheng Yang, Kaisen Yang, Baiting Wu, Xun Zhang, Qianrui Yang, Haotong Qin, He Zhang, and Yulun Zhang. 2025b. Treeq: Pushing the quantization boundary of diffusion transformer via tree-structured mixed-precision search. *arXiv preprint arXiv:2512.06353*.

Kaicheng Yang, Xun Zhang, Haotong Qin, Yucheng Lin, Kaisen Yang, Xianglong Yan, and Yulun Zhang. 2025c. Robuq: Pushing dits to w1. 58a2 via robust activation quantization. *arXiv preprint arXiv:2509.23582*.

Shenghao Ye, Yu Guo, Dong Jin, Yuxiang Wang, Yikai Shen, Yunpeng Hou, Shuangwu Chen, Jianyang, and Xiaofeng Jiang. 2026a. When TableQA meets noise: A dual denoising framework for complex questions and large-scale tables. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24022–24045, San Diego, California, United States. Association for Computational Linguistics.

Shenghao Ye, Yu Guo, Zhengheng Li, Shuangwu Chen, and Jian Yang. 2026b. Rubric-guided process reward for stepwise model routing. *Preprint*, arXiv:2605.29310.

Shenghao Ye, Yuxiang Wang, Yu Guo, Dong Jin, Shuangwu Chen, and Jian Yang. 2026c. Rethinking stepwise model routing: A cost-efficient table reasoning perspective. *Preprint*, arXiv:2605.29319.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *Preprint*, arXiv:1905.07830.

Hengyuan Zhang, Xinrong Chen, Zunhai Su, Xiao Liang, Jing Xiong, Wendong Xu, He Xiao, Chaofan Tao, Wei Zhang, Ruobing Xie, and 1 others. 2026a. Beyond outliers: A data-free layer-wise mixed-precision quantization approach driven by numerical and structural dual-sensitivity. *arXiv preprint arXiv:2603.17354*.

Xun Zhang, Kaicheng Yang, Hongliang Lu, Haotong Qin, Yong Guo, and Yulun Zhang. 2026b. Q-dit4sr: Exploration of detail-preserving diffusion transformer quantization for real-world image super-resolution. *arXiv preprint arXiv:2602.01273*.

Junchen Zhao, Ali Derakhshan, Dushyant Bharadwaj, Jayden Kana Hyman, Junhao Dong, Sangeetha Abdu Jyothi, and Ian Harris. 2025a. Impq: Interaction-aware layerwise mixed precision quantization for llms. *arXiv e-prints*, pages arXiv–2509.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. 2024. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209.

Zhixiong Zhao, Haomin Li, Fangxin Liu, Yuncheng Lu, Zongwu Wang, Tao Yang, Li Jiang, and Haibing Guan. 2025b. Quark: Quantization-enabled circuit sharing for transformer acceleration by exploiting common patterns in nonlinear operations. In *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE.

Zhixiong Zhao, Fangxin Liu, Junjie Wang, Chenyang Guan, Zongwu Wang, Li Jiang, and Haibing Guan. 2026a. Specquant: Spectral decomposition and adaptive truncation for ultra-low-bit llms quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 28786–28794.

Zhixiong Zhao, Zukang Xu, Zhixuan Chen, Xing Hu, Zhe Jiang, and Dawei Yang. 2026b. Twla: Achieving ternary weights and low-bit activations for llms via post-training quantization. *arXiv preprint arXiv:2606.13054*.

Zhixiong Zhao, Zukang Xu, and Dawei Yang. 2026c. Bwla: Breaking the barrier of w1ax post-training quantization for llms. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19264–19290.

Chunzheng Zhu, Jianxin Lin, Guanghua Tan, Ningbo Zhu, Kenli Li, Chunlian Wang, and Shengli Li. 2024. Advancing ultrasound medical continuous learning with task-specific generalization and adaptability. In *2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 3019–3025. IEEE.

Chunzheng Zhu, Yangfang Lin, Shen Chen, Yijun Wang, and Jianxin Lin. 2026a. Medeyes: Learning dynamic visual focus for medical progressive diagnosis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 13916–13924.

Chunzheng Zhu, Yangfang Lin, Jialin Shao, Jianxin Lin, and Yijun Wang. 2025a. Pathology-aware prototype evolution via llm-driven semantic disambiguation for multicenter diabetic retinopathy diagnosis. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 9196–9205.

Chunzheng Zhu, Jialin Shao, Jianxin Lin, Yijun Wang, Jing Wang, Jinhui Tang, and Kenli Li. 2025b. fmri2ges: Co-speech gesture reconstruction from fmri signal with dual brain decoding alignment. *IEEE Transactions on Circuits and Systems for Video Technology*.

Chunzheng Zhu, Yijun Wang, Jianxin Lin, Feng Wang, Hongwei Wang, Lei Zhao, Shengli Li, and Kenli Li. 2026b. Anatomy-anchored self-supervision: Distilling vision foundation models for invariant ultrasound representation. *MICCAI 2026 (Early Accept)*.

Chunzheng Zhu, Jiaqi Zeng, Junyu Jiang, Jianxin Lin, and Yijun Wang. 2026c. Medsynapse-v: Bridging visual perception and clinical intuition via latent memory evolution. *Preprint*, arXiv:2604.26283.

Libo Zhu, Haotong Qin, Kaicheng Yang, Wenbo Li, Yong Guo, Yulun Zhang, Susanto Rahardja, and Xiaokang Yang. 2025c. Qartsr: Quantization via reverse-module and timestep-retraining in one-step diffusion based image super-resolution. *arXiv preprint arXiv:2503.05584*.

A Appendix

Appendix Overview

- Section A.1: Detailed Algorithms of AF1.
  - Section A.1.1: Overall Workflow of AF1.
  - Section A.1.2: Null-space-Aware Binary Factorization.
  - Section A.1.3: Hierarchical Shapley Allocation.
  - Section A.1.4: ADMM-SVID Framework in DBF.
- Section A.2: Detailed Proofs.
  - Section A.2.1: Representation Capacity of Double-Factor Binary Parameterization.
  - Section A.2.2: Equivalence Between the Null Eigenspace of $\mathbf{B}\mathbf{B}^\top$ and the Left Null Space of $\mathbf{B}$.
  - Section A.2.3: Adaptive Residual-Energy Cutoff.
  - Section A.2.4: Proof of the Approximate Null-Space Projection Property.
  - Section A.2.5: Closed-form Solution for Scale Compensation.
  - Section A.2.6: Interaction-aware Property of Shapley Allocation.
  - Section A.2.7: Dual Distortion Decomposition for Block-Level Allocation.
  - Section A.2.8: Motivation for Size-Aware Sensitivity Density.
- Section A.3: Effective Bit-Width of Binary PTQ Methods.
- Section A.4: More Experimental Results.
  - Section A.4.1: More Detailed Results.
  - Section A.4.2: Ablation on the Residual-Energy Threshold.
  - Section A.4.3: Ablation on Allocation-Sensitivity Metrics.
  - Section A.4.4: Additional Ablation on HiSA Hyperparameters.
  - Section A.4.5: Ablation on Calibration Data.
- Section A.5: Dialog Examples.
- Section A.6: Use of Large Language Models.

A.1 Detailed Algorithms of AF1

This appendix provides the algorithmic workflow of AF1. We present three pseudocode blocks corresponding to the overall PTQ pipeline, Null-space-Aware Binary Factorization (NABF), and Hierarchical Shapley Allocation (HiSA), respectively.

A.1.1 Overall Workflow of AF1

Pipeline description. AF1 combines hierarchical structural allocation with operator-level binary reconstruction. Given a pretrained model and a calibration set, AF1 first collects the statistics required by HiSA and NABF. HiSA determines the layer-wise and module-wise structural budgets, while NABF reconstructs each target linear operator under the assigned intermediate dimension. After all target operators are replaced, AF1 freezes the binary sign matrices and performs scale-only global reconstruction. The overall procedure is summarized in Algorithm 1.

A.1.2 Null-space-Aware Binary Factorization

Hessian-aware surrogate space. NABF operates on each target linear operator under the intermediate dimension allocated by HiSA. It first constructs a Hessian-aware surrogate weight by scaling the original weight with input-activation and output-gradient statistics. This converts the Hessian-weighted reconstruction problem into a Frobenius reconstruction problem in the surrogate space, while retaining task-sensitive channel information.

Binary factorization and null-space compensation. After surrogate reparameterization, NABF uses a double-factor binary structure with two binary sign matrices and several continuous scale vectors. The continuous factors are updated by alternating ADMM steps and then projected back to the binary-scaled domain through SVID. Since SVID projection introduces discrete residuals, NABF constructs an approximate null-space projector from the fixed factor and folds the compensated residual into the existing intermediate scaling vectors. The complete procedure is given in Algorithm 2.

Compact operators. To keep the single-column pseudocode readable, we use several compact operators. $\mathrm{ADMMLeft}$ and $\mathrm{ADMMRight}$ denote the closed-form ADMM proxy updates in the main text. $\mathrm{NullProj}(\mathbf{C}, \delta)$ returns the projector spanned by eigenvectors of $\mathbf{C}$ whose eigenvalues are below δ. $\mathrm{ColScaleFit}(\mathbf{X}, \mathbf{Y})$ solves the

Algorithm 1: AF1: Overall PTQ Pipeline

Input: Pretrained LLM M; calibration set $\mathcal{D}_{\text{cal}}$; total budget B_{total}; original dimension r_{orig}; proxy dimension r_{proxy}
Output: Compressed model $\hat{M}$
Collect calibration statistics from M on $\mathcal{D}_{\text{cal}}$;
$\{B_l\}, \{b_i\} \leftarrow \text{HiSA}(M, \mathcal{D}_{\text{cal}}, B_{\text{total}}, r_{\text{orig}}, r_{\text{proxy}})$;
for *each target linear operator* $\mathbf{W}_i$ **do**
  $r_i \leftarrow \text{RankFromBudget}(b_i, \mathbf{W}_i)$;
  $\hat{\mathbf{W}}_i \leftarrow \text{NABF}(\mathbf{W}_i, r_i, \mathcal{D}_{\text{cal}})$;
  Replace $\mathbf{W}_i$ in M with $\hat{\mathbf{W}}_i$;
Freeze all binary sign matrices;
Collect learnable scale vectors $\mathcal{S}$;
Optimize only $\mathcal{S}$ using the KL objective in Eq. 11;
Assemble the final compressed model $\hat{M}$;
return $\hat{M}$

column-wise least-squares scaling from $\mathbf{X}$ to $\mathbf{Y}$, and $\text{RowScaleFit}(\mathbf{X}, \mathbf{Y})$ is its row-wise counterpart.

A.1.3 Hierarchical Shapley Allocation

Layer-level allocation. HiSA first estimates the global structural sensitivity of each Transformer layer. Instead of evaluating layers independently, it progressively degrades layers from the original intermediate dimension to a proxy dimension under sampled permutations, and measures the resulting change in end-to-end NLL. This gives a layer-level Shapley estimate that reflects cross-layer interactions and error propagation.

Block-level allocation. Given the layer budget, HiSA further assigns capacity to submodules within each layer. For each sampled submodule permutation, HiSA progressively degrades submodules and measures the distortion of the layer output. Directional and magnitude distortions are estimated separately, normalized by MAD, and fused by a Soft-OR rule. The fused score is converted into a size-aware density and then into a module-wise budget. The complete allocation procedure is shown in Algorithm 3.

Budget-to-rank conversion. After HiSA returns module-wise budgets, AF1 converts each budget into the intermediate dimension used by NABF. For a target matrix $\mathbf{W}_i \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the dominant binary storage of the double-factor representation is $r_i(d_{\text{out}} + d_{\text{in}})$ bits. Thus, the practical rank can be computed as

$$r_i = \text{Round}\left(\frac{b_i d_{\text{out}} d_{\text{in}}}{d_{\text{out}} + d_{\text{in}}}\right), \tag{20}$$

with clipping to a feasible range in implementation.

A.1.4 ADMM-SVID Framework in DBF

Double-binary factorization objective. NABF builds upon the double-binary factorization framework introduced in DBF (Boža and Macko, 2025). Given a surrogate weight matrix $\mathbf{W}'$, the goal is to approximate it by two binary-scaled factors:

$$\min \quad \left\| \mathbf{W}' - \left(\mathbf{a} \odot \mathbf{A}_{\pm 1} \odot \mathbf{m}^\top\right)\left(\mathbf{B}_{\pm 1} \odot \mathbf{b}^\top\right) \right\|_F^2 . \tag{21}$$

Following DBF, the middle scaling vector is split into two parts, which can be merged back after optimization:

$$\min \quad \left\| \mathbf{W}' - \left(\mathbf{a} \odot \mathbf{A}_{\pm 1} \odot \mathbf{m}_1^\top\right)\left(\mathbf{m}_2 \odot \mathbf{B}_{\pm 1} \odot \mathbf{b}^\top\right) \right\|_F^2 . \tag{22}$$

We denote the two scaled factors as

$$\begin{aligned} \mathbf{A} &= \mathbf{a} \odot \mathbf{A}_{\pm 1} \odot \mathbf{m}_1^\top, \\ \mathbf{B} &= \mathbf{m}_2 \odot \mathbf{B}_{\pm 1} \odot \mathbf{b}^\top. \end{aligned} \tag{23}$$

The factorization is then optimized by alternating minimization: fixing $\mathbf{B}$ to update $\mathbf{A}$, and fixing $\mathbf{A}$ to update $\mathbf{B}$.

ADMM subproblem. When updating the left factor with $\mathbf{B}$ fixed, the constrained subproblem is

$$\begin{aligned} \min_{\mathbf{A}} \quad & \left\| \mathbf{AB} - \mathbf{W}' \right\|_F^2 , \\ \text{s.t.} \quad & \mathbf{A} = \mathbf{a} \odot \mathbf{A}_{\pm 1} \odot \mathbf{m}_1^\top . \end{aligned} \tag{24}$$

Algorithm 2: NABF: Null-space-Aware Binary Factorization

Input: Weight $\mathbf{W}$; calibration activations $\mathbf{x}$; output gradients $\mathbf{g}$; intermediate dimension r; ADMM penalty ρ; max iterations T; threshold δ

Output: Binary factors $\mathbf{A}_{\pm 1}, \mathbf{B}_{\pm 1}$ and scales $\mathbf{a}, \mathbf{m}_1, \mathbf{m}_2, \mathbf{b}$

Compute $i_j = \sqrt{\mathbb{E}[x_j^2]}$ and $o_u = \sqrt{\mathbb{E}[g_u^2]}$;
Construct $\mathbf{W}' = \mathbf{o} \odot \mathbf{W} \odot \mathbf{i}^\top$;
Initialize $\mathbf{A}_{\pm 1}, \mathbf{B}_{\pm 1}, \mathbf{a}, \mathbf{m}_1, \mathbf{m}_2, \mathbf{b}$ by SVID or low-rank approximation;
Form scaled factors $\mathbf{A}$ and $\mathbf{B}$;
Initialize ADMM dual variables $\mathbf{U}_A$ and $\mathbf{U}_B$;
for $t = 1$ **to** T **do**
  `// Left-factor update`
  $\hat{\mathbf{A}} \leftarrow \mathrm{ADMMLeft}(\mathbf{W}', \mathbf{A}, \mathbf{B}, \mathbf{U}_A, \rho)$;
  $\mathbf{A}_q \leftarrow \mathrm{SVID}(\hat{\mathbf{A}} + \mathbf{U}_A)$;
  $\mathbf{E}_A \leftarrow \hat{\mathbf{A}} - \mathbf{A}_q$;
  $\mathbf{P}_B \leftarrow \mathrm{NullProj}(\mathbf{B}\mathbf{B}^\top, \delta)$;
  $\mathbf{A}_{\text{target}} \leftarrow \hat{\mathbf{A}} - \mathbf{E}_A \mathbf{P}_B$;
  $\boldsymbol{\gamma}^* \leftarrow \mathrm{ColScaleFit}(\mathbf{A}_q, \mathbf{A}_{\text{target}})$;
  $\mathbf{m}_1 \leftarrow \mathbf{m}_1 \odot \boldsymbol{\gamma}^*$;
  Update scaled factor $\mathbf{A}$;
  `// Right-factor update`
  $\hat{\mathbf{B}} \leftarrow \mathrm{ADMMRight}(\mathbf{W}', \mathbf{A}, \mathbf{B}, \mathbf{U}_B, \rho)$;
  $\mathbf{B}_q \leftarrow \mathrm{SVID}(\hat{\mathbf{B}} + \mathbf{U}_B)$;
  $\mathbf{E}_B \leftarrow \hat{\mathbf{B}} - \mathbf{B}_q$;
  $\mathbf{P}_A \leftarrow \mathrm{NullProj}(\mathbf{A}^\top \mathbf{A}, \delta)$;
  $\mathbf{B}_{\text{target}} \leftarrow \hat{\mathbf{B}} - \mathbf{P}_A \mathbf{E}_B$;
  $\boldsymbol{\eta}^* \leftarrow \mathrm{RowScaleFit}(\mathbf{B}_q, \mathbf{B}_{\text{target}})$;
  $\mathbf{m}_2 \leftarrow \boldsymbol{\eta}^* \odot \mathbf{m}_2$;
  Update scaled factor $\mathbf{B}$;
  Update ADMM dual variables $\mathbf{U}_A$ and $\mathbf{U}_B$;
return $\mathbf{A}_{\pm 1}, \mathbf{B}_{\pm 1}, \mathbf{a}, \mathbf{m}_1, \mathbf{m}_2, \mathbf{b}$

This constraint is non-convex because of the binary sign matrix. DBF solves this subproblem using ADMM (Neal et al., 2011). For a generic constrained problem

$$\min_{\mathbf{X}} f(\mathbf{X}), \qquad \text{s.t. } \mathbf{X} \in \mathcal{C}, \tag{25}$$

one ADMM iteration is

$$\begin{aligned} \mathbf{X}^{(k+1)} &= \arg\min_{\mathbf{X}} f(\mathbf{X}) + \frac{\rho}{2} \left\| \mathbf{X} - \mathbf{Z}^{(k)} + \mathbf{U}^{(k)} \right\|_F^2, \\ \mathbf{Z}^{(k+1)} &= \Pi_{\mathcal{C}} \left(\mathbf{X}^{(k+1)} + \mathbf{U}^{(k)} \right), \\ \mathbf{U}^{(k+1)} &= \mathbf{U}^{(k)} + \mathbf{X}^{(k+1)} - \mathbf{Z}^{(k+1)}, \end{aligned} \tag{26}$$

where ρ is the penalty coefficient, $\mathbf{U}$ is the scaled dual variable, and $\Pi_{\mathcal{C}}$ denotes Euclidean projection onto the feasible set.

SVID projection. In DBF, the feasible set $\mathcal{C}$ consists of matrices that can be written as $\mathbf{a} \odot \mathbf{A}_{\pm 1} \odot \mathbf{m}_1^\top$. The projection $\Pi_{\mathcal{C}}$ is implemented by the SVID projection (Xu et al., 2024). Given a matrix $\mathbf{Z}$, SVID first sets

$$\mathbf{Z}_{\pm 1} = \mathrm{Sign}(\mathbf{Z}), \tag{27}$$

and then approximates the magnitude matrix by a rank-1 factorization:

$$|\mathbf{Z}| \approx \mathbf{a}\mathbf{m}_1^\top. \tag{28}$$

The projected matrix is therefore

$$\mathrm{SVID}(\mathbf{Z}) = \mathbf{a} \odot \mathbf{Z}_{\pm 1} \odot \mathbf{m}_1^\top. \tag{29}$$

In practice, the rank-1 magnitude approximation can be efficiently computed by power iteration.

Algorithm 3: HiSA: Hierarchical Shapley Allocation

Input: Model M; calibration set $\mathcal{D}_{\text{cal}}$; layer set T; submodules $\mathcal{N}_l$; total budget B_{total}; dimensions $r_{\text{orig}}, r_{\text{proxy}}$; temperature τ; smoothing exponent α; permutation numbers $M_{\text{layer}}, M_{\text{block}}$

Output: Layer budgets $\{B_l\}$ and module budgets $\{b_i\}$

Initialize $\Phi_l = 0$ for all layers;

for $m = 1$ **to** M_{layer} **do**

  Sample a random layer permutation π_m;

  Initialize $S = T$ and evaluate $v_{\text{NLL}}(S)$;

  for *each layer l following π_m* **do**

    Degrade l from r_{orig} to r_{proxy};

    Evaluate $v_{\text{NLL}}(S \setminus \{l\})$;

    $\Phi_l \leftarrow \Phi_l + v_{\text{NLL}}(S \setminus \{l\}) - v_{\text{NLL}}(S)$;

    $S \leftarrow S \setminus \{l\}$;

$\Phi_l \leftarrow \Phi_l / M_{\text{layer}}$ for all layers;

$\{B_l\} \leftarrow \text{SoftmaxBudget}(\{\Phi_l\}, B_{\text{total}}, \tau)$;

for *each layer $l = 1, \ldots, L$* **do**

  Compute reference output H_l^{ref};

  Initialize $\Phi_{\text{dir}}^{(i)} = 0$ and $\Phi_{\text{mag}}^{(i)} = 0$ for all $i \in \mathcal{N}_l$;

  for $m = 1$ **to** M_{block} **do**

    Sample a random submodule permutation π_m;

    Initialize $S_0 = \varnothing$, $\hat{E}_{\text{dir}}(S_0) = 0$, and $\hat{E}_{\text{mag}}(S_0) = 0$;

    for *each submodule i following π_m* **do**

      Degrade i to the low-capacity state and compute $H_l(S_t)$;

      $E_{\text{dir}}(S_t) \leftarrow r_{\text{dir}}(H_l^{\text{ref}}, H_l(S_t))$;

      $E_{\text{mag}}(S_t) \leftarrow r_{\text{mag}}(H_l^{\text{ref}}, H_l(S_t))$;

      $\hat{E}_c(S_t) \leftarrow \max(E_c(S_t), \hat{E}_c(S_{t-1}))$, $c \in \{\text{dir}, \text{mag}\}$;

      $\Phi_c^{(i)} \leftarrow \Phi_c^{(i)} + \hat{E}_c(S_t) - \hat{E}_c(S_{t-1})$, $c \in \{\text{dir}, \text{mag}\}$;

  $\Phi_c^{(i)} \leftarrow \Phi_c^{(i)} / M_{\text{block}}$, $c \in \{\text{dir}, \text{mag}\}$;

  $z_c^{(i)} \leftarrow \text{MADNorm}(\Phi_c^{(i)})$, $c \in \{\text{dir}, \text{mag}\}$;

  $P_c^{(i)} \leftarrow 1/(1 + \exp(-z_c^{(i)}))$;

  $v_{\text{base}}^{(i)} \leftarrow 1 - (1 - P_{\text{dir}}^{(i)})(1 - P_{\text{mag}}^{(i)})$;

  $\omega_i \leftarrow \left(v_{\text{base}}^{(i)} / (|\mathbf{W}_i| + \epsilon) + \epsilon \right)^{\alpha}$;

  $b_i \leftarrow \text{ModuleBudget}(B_l, \omega_i, |\mathbf{W}_i|)$ for all $i \in \mathcal{N}_l$;

return $\{B_l\}, \{b_i\}$

Closed-form proxy updates. For the left-factor update, applying ADMM to the above subproblem gives the continuous proxy

$$\hat{\mathbf{A}}^{(k+1)} = \left(\mathbf{W}'\mathbf{B}^\top + \rho \left(\mathbf{A}^{(k)} - \mathbf{U}^{(k)} \right) \right) \left(\mathbf{B}\mathbf{B}^\top + \rho \mathbf{I} \right)^{-1}, \tag{30}$$

followed by SVID projection and dual update:

$$\begin{aligned} \mathbf{A}^{(k+1)} &= \text{SVID}\left(\hat{\mathbf{A}}^{(k+1)} + \mathbf{U}^{(k)} \right), \\ \mathbf{U}^{(k+1)} &= \mathbf{U}^{(k)} + \hat{\mathbf{A}}^{(k+1)} - \mathbf{A}^{(k+1)}. \end{aligned} \tag{31}$$

The right-factor update is derived symmetrically:

$$\hat{\mathbf{B}}^{(k+1)} = \left(\mathbf{A}^\top \mathbf{A} + \rho \mathbf{I} \right)^{-1} \left(\mathbf{A}^\top \mathbf{W}' + \rho \left(\mathbf{B}^{(k)} - \mathbf{U}^{(k)} \right) \right), \tag{32}$$

followed by SVID projection onto the right-factor binary-scaled structure. NABF uses this ADMM-SVID procedure as the baseline discrete optimization framework, and then introduces null-space-aware compensation to suppress the projection residual left by SVID.

A.2 Detailed Proofs

A.2.1 Representation Capacity of Double-Factor Binary Parameterization

We provide a justification for why the double-factor binary parameterization used in NABF has higher representation capacity than a single binary-scaled matrix. The argument follows the general motivation of double-factor compression in DBF/DSF (Boža and Macko, 2025): instead of approximating a weight matrix by one constrained matrix, representing it as the product of two constrained factors introduces intermediate paths that increase expressiveness under a controlled structural budget.

Single-factor binary-scaled representation. A standard SVID-style binary-scaled matrix can be written as

$$\widehat{\mathbf{W}}_{\text{single}} = \mathbf{a} \odot \mathbf{S} \odot \mathbf{b}^\top = \operatorname{Diag}(\mathbf{a})\mathbf{S}\operatorname{Diag}(\mathbf{b}), \quad \mathbf{S} \in \{-1,+1\}^{n\times m}. \tag{33}$$

Although the sign matrix $\mathbf{S}$ is fully binary and flexible, the magnitude part is separable:

$$\left|\widehat{\mathbf{W}}_{\text{single},ij}\right| = |a_i||b_j|. \tag{34}$$

Therefore, the magnitude matrix of a single-factor binary-scaled representation is rank one. Equivalently, for any two rows i_1, i_2 and two columns j_1, j_2, it must satisfy the multiplicative consistency constraint

$$\left|\widehat{\mathbf{W}}_{\text{single},i_1j_1}\right|\left|\widehat{\mathbf{W}}_{\text{single},i_2j_2}\right| = \left|\widehat{\mathbf{W}}_{\text{single},i_1j_2}\right|\left|\widehat{\mathbf{W}}_{\text{single},i_2j_1}\right|. \tag{35}$$

This constraint limits the ability of a single binary-scaled matrix to fit heterogeneous magnitude patterns, which are common in LLM weights after Hessian-aware reparameterization.

Double-factor binary representation. NABF instead uses

$$\widehat{\mathbf{W}}_{\text{double}} = \left(\mathbf{a} \odot \mathbf{A}_{\pm1} \odot \mathbf{m}_1^\top\right)\left(\mathbf{m}_2 \odot \mathbf{B}_{\pm1} \odot \mathbf{b}^\top\right), \tag{36}$$

where $\mathbf{A}_{\pm1} \in \{-1,+1\}^{n\times r}$ and $\mathbf{B}_{\pm1} \in \{-1,+1\}^{r\times m}$. Let

$$\boldsymbol{\lambda} = \mathbf{m}_1 \odot \mathbf{m}_2. \tag{37}$$

Then the double-factor form can be equivalently written as

$$\widehat{\mathbf{W}}_{\text{double}} = \operatorname{Diag}(\mathbf{a})\mathbf{A}_{\pm1}\operatorname{Diag}(\boldsymbol{\lambda})\mathbf{B}_{\pm1}\operatorname{Diag}(\mathbf{b}). \tag{38}$$

Thus, each entry is given by

$$\widehat{\mathbf{W}}_{\text{double},ij} = a_i b_j \sum_{k=1}^{r} \lambda_k A_{\pm1,ik} B_{\pm1,kj}. \tag{39}$$

Equation 39 shows that each input-output connection is no longer determined by a single binary sign and a separable magnitude term. Instead, it is the sum of r binary paths through the intermediate dimension. The path coefficients λ_k provide continuous degrees of freedom, while $\mathbf{A}_{\pm1}$ and $\mathbf{B}_{\pm1}$ keep the dominant parameters binary.

Strictly richer magnitude patterns. The double-factor form can represent magnitude patterns that violate the rank-one constraint in Eq. 35. Consider a 2×2 example with $\mathbf{a} = \mathbf{b} = \mathbf{1}$, $r = 2$,

$$\mathbf{A}_{\pm1} = \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}, \quad \mathbf{B}_{\pm1} = \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{bmatrix}1\\ 1/2\end{bmatrix}. \tag{40}$$

Then

$$\mathbf{A}_{\pm1}\operatorname{Diag}(\boldsymbol{\lambda})\mathbf{B}_{\pm1} = \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 1/2\end{bmatrix}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \begin{bmatrix}3/2 & 1/2\\ 1/2 & 3/2\end{bmatrix}. \tag{41}$$

The magnitude matrix in Eq. 41 has determinant

$$\frac{3}{2}\cdot\frac{3}{2} - \frac{1}{2}\cdot\frac{1}{2} = 2 \neq 0. \tag{42}$$

Hence, it is not rank one and cannot be represented by any single-factor binary-scaled form $\operatorname{Diag}(\mathbf{a})\mathbf{S}\operatorname{Diag}(\mathbf{b})$, whose magnitude matrix must satisfy Eq. 35. This example shows that even with $r = 2$, the double-factor form can express non-separable magnitude structures that are impossible for a single binary-scaled matrix.

Path-sum interpretation. The capacity gain can also be understood from Eq. 39. For a fixed pair (i, j), the normalized entry

$$\frac{\widehat{\mathbf{W}}_{\text{double},ij}}{a_i b_j} = \sum_{k=1}^{r} \lambda_k A_{\pm1,ik} B_{\pm1,kj} \tag{43}$$

is a signed combination of r path weights. With distinct λ_k, different input-output pairs can realize different sums depending on the binary path signs. Therefore, the model can produce multiple effective magnitude levels and correlation patterns, instead of being restricted to the separable magnitude $|a_i||b_j|$ of a single-factor representation.

Implication for NABF. The above analysis explains why NABF adopts a double-factor binary parameterization before applying null-space compensation. The double-factor structure increases the representational capacity available under a genuine 1-bit storage budget by introducing an intermediate binary path space. At the same time, the dominant matrices $\mathbf{A}_{\pm 1}$ and $\mathbf{B}_{\pm 1}$ remain binary, and the additional continuous parameters are limited to lightweight scale vectors. This provides a better starting point for ADMM-SVID projection and allows the subsequent null-space-aware scale recalibration to absorb discrete projection errors without introducing dense correction matrices.

A.2.2 Equivalence Between the Null Eigenspace of $\mathbf{B}\mathbf{B}^\top$ and the Left Null Space of $\mathbf{B}$

We prove that the null eigenspace of the uncentered covariance matrix $\mathbf{B}\mathbf{B}^\top$ is equivalent to the left null space of $\mathbf{B}$. This justifies constructing the approximate null-space projector from $\mathbf{C}_B = \mathbf{B}\mathbf{B}^\top$ in NABF.

Let $\mathbf{B} \in \mathbb{R}^{r\times d}$ and define

$$\mathbf{C}_B = \mathbf{B}\mathbf{B}^\top \in \mathbb{R}^{r\times r}. \tag{44}$$

For any vector $\mathbf{z} \in \mathbb{R}^r$, we have

$$\begin{aligned} \mathbf{z}^\top \mathbf{C}_B \mathbf{z} &= \mathbf{z}^\top \mathbf{B}\mathbf{B}^\top \mathbf{z} \\ &= \left(\mathbf{B}^\top \mathbf{z}\right)^\top \left(\mathbf{B}^\top \mathbf{z}\right) \\ &= \left\|\mathbf{B}^\top \mathbf{z}\right\|_2^2. \end{aligned} \tag{45}$$

Since $\mathbf{C}_B$ is positive semidefinite, $\mathbf{z}$ belongs to the null eigenspace of $\mathbf{C}_B$ if and only if the above quadratic form is zero:

$$\begin{aligned} \mathbf{z} \in \mathrm{Null}(\mathbf{C}_B) &\iff \mathbf{z}^\top \mathbf{C}_B \mathbf{z} = 0 \\ &\iff \left\|\mathbf{B}^\top \mathbf{z}\right\|_2^2 = 0 \\ &\iff \mathbf{B}^\top \mathbf{z} = \mathbf{0}. \end{aligned} \tag{46}$$

Therefore,

$$\mathrm{Null}\left(\mathbf{B}\mathbf{B}^\top\right) = \mathrm{Null}\left(\mathbf{B}^\top\right). \tag{47}$$

That is, the zero-eigenvalue eigenspace of $\mathbf{B}\mathbf{B}^\top$ is exactly the left null space of $\mathbf{B}$.

Let $\widehat{\mathbf{U}}$ be the eigenvectors of $\mathbf{C}_B$ corresponding to zero eigenvalues, and define

$$\mathbf{P}_B = \widehat{\mathbf{U}}\widehat{\mathbf{U}}^\top. \tag{48}$$

Then $\mathbf{P}_B$ is the orthogonal projector onto $\mathrm{Null}(\mathbf{B}^\top)$. For any vector $\mathbf{v}$, $\mathbf{P}_B\mathbf{v} \in \mathrm{Null}(\mathbf{B}^\top)$, and thus

$$\mathbf{B}^\top \mathbf{P}_B \mathbf{v} = \mathbf{0}. \tag{49}$$

Equivalently, this implies

$$\mathbf{P}_B \mathbf{B} = \mathbf{0}. \tag{50}$$

Hence, projecting a residual component onto this subspace makes it vanish after multiplication by the fixed factor $\mathbf{B}$:

$$\left(\mathbf{E}_A \mathbf{P}_B\right) \mathbf{B} = \mathbf{E}_A \left(\mathbf{P}_B \mathbf{B}\right) = \mathbf{0}. \tag{51}$$

This property is the basis of the null-space compensation used in NABF.

In practice, due to numerical noise and the finite intermediate dimension, the eigenvalues of $\mathbf{C}_B$ are rarely exactly zero. Therefore, we use eigenvectors associated with zero or near-zero eigenvalues to construct an approximate projector $\mathbf{P}_B$, which satisfies

$$\mathbf{P}_B \mathbf{B} \approx \mathbf{0}. \tag{52}$$

This approximate property is sufficient for suppressing the propagated projection residual $\mathbf{E}_A\mathbf{B}$.

Symmetric case for the right-factor update. The same argument applies when updating the right factor. Given a fixed left factor $\mathbf{A} \in \mathbb{R}^{n\times r}$, define

$$\mathbf{C}_A = \mathbf{A}^\top \mathbf{A} \in \mathbb{R}^{r\times r}. \tag{53}$$

For any $\mathbf{z} \in \mathbb{R}^r$,

$$\mathbf{z}^\top \mathbf{C}_A \mathbf{z} = \left\|\mathbf{A}\mathbf{z}\right\|_2^2. \tag{54}$$

Thus,

$$\mathrm{Null}\left(\mathbf{A}^\top \mathbf{A}\right) = \mathrm{Null}(\mathbf{A}). \tag{55}$$

If $\mathbf{P}_A$ is constructed from the zero or near-zero eigenspace of $\mathbf{A}^\top\mathbf{A}$, then

$$\mathbf{A}\mathbf{P}_A \approx \mathbf{0}. \tag{56}$$

Consequently, for the right-factor projection residual $\mathbf{E}_B$, the compensated component satisfies

$$\mathbf{A}\left(\mathbf{P}_A \mathbf{E}_B\right) = \left(\mathbf{A}\mathbf{P}_A\right) \mathbf{E}_B \approx \mathbf{0}. \tag{57}$$

This gives the symmetric null-space compensation used for the right-factor update.

A.2.3 Adaptive Residual-Energy Cutoff

We describe how the cutoff index used in the adaptive residual-energy criterion is determined. Given the decomposition

$$\mathbf{C}_B = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top, \quad \mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_r), \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \geq 0, \tag{58}$$

the leading eigenspace captures most of the spectral energy, while the trailing eigenspace contains the remaining low-energy directions. A fixed absolute eigenvalue threshold is sensitive to the scale of $\mathbf{C}_B$, which may vary across layers and submodules. We therefore determine the cutoff by the fraction of residual spectral energy.

For a candidate cutoff index k, we define the total and residual spectral energies as

$$E_{\mathrm{tot}} = \sum_{i=1}^{r} \lambda_i, \qquad E_{\mathrm{res}}(k) = \sum_{i=k+1}^{r} \lambda_i. \tag{59}$$

The residual-energy fraction is then

$$\rho_k = \frac{E_{\mathrm{res}}(k)}{E_{\mathrm{tot}} + \epsilon} = \frac{\sum_{i=k+1}^{r} \lambda_i}{\sum_{i=1}^{r} \lambda_i + \epsilon}, \tag{60}$$

where ϵ is a small constant for numerical stability. We choose the first index k satisfying

$$\rho_k \leq \eta, \tag{61}$$

where η controls the maximum fraction of spectral energy assigned to the trailing subspace. Equivalently, the retained eigenspace explains at least $1 - \eta$ of the total spectral energy. The selected approximate null-space basis is then

$$\widehat{\mathbf{U}} = \mathbf{U}_{:,k+1:r}. \tag{62}$$

This criterion is scale-invariant: multiplying $\mathbf{C}_B$ by any positive constant scales both E_{tot} and $E_{\mathrm{res}}(k)$ by the same factor, leaving ρ_k unchanged up to the numerical stabilizer ϵ. Therefore, the cutoff adapts to the spectrum of each factor covariance matrix and avoids manually choosing a global absolute eigenvalue threshold across layers and submodules.

A.2.4 Proof of the Approximate Null-Space Projection Property

We prove that the projector constructed from near-zero singular vectors satisfies $\mathbf{P}_B\mathbf{B} \approx \mathbf{0}$. Let $\widehat{\mathbf{U}} = [\mathbf{u}_1, \ldots, \mathbf{u}_r]$ collect the singular vectors of $\mathbf{C}_B = \mathbf{B}\mathbf{B}^\top$ whose singular values are zero or near-zero. The projector is defined as

$$\mathbf{P}_B = \widehat{\mathbf{U}}\widehat{\mathbf{U}}^\top. \tag{63}$$

For each selected vector $\mathbf{u}_k$, the near-zero singular value condition gives

$$\mathbf{C}_B\mathbf{u}_k \approx \mathbf{0}. \tag{64}$$

Since $\mathbf{C}_B = \mathbf{B}\mathbf{B}^\top$ is positive semi-definite, we have

$$\mathbf{u}_k^\top \mathbf{C}_B \mathbf{u}_k = \mathbf{u}_k^\top \mathbf{B}\mathbf{B}^\top \mathbf{u}_k = \|\mathbf{B}^\top \mathbf{u}_k\|_2^2 \approx 0. \tag{65}$$

Thus, $\mathbf{B}^\top \mathbf{u}_k \approx \mathbf{0}$, or equivalently $\mathbf{u}_k^\top \mathbf{B} \approx \mathbf{0}$. Stacking all selected vectors yields

$$\widehat{\mathbf{U}}^\top \mathbf{B} \approx \mathbf{0}. \tag{66}$$

Therefore,

$$\mathbf{P}_B\mathbf{B} = \widehat{\mathbf{U}}\widehat{\mathbf{U}}^\top \mathbf{B} \approx \mathbf{0}. \tag{67}$$

This shows that $\mathbf{P}_B$ approximately projects onto the left null space of $\mathbf{B}$, which justifies its use for steering projection errors into directions that minimally affect the product with $\mathbf{B}$.

A.2.5 Closed-form Solution for Scale Compensation

We provide the derivation of the closed-form solution used for scale compensation in NABF. After null-space compensation, the continuous target $\mathbf{A}_{\mathrm{target}}$ preserves the propagated behavior of the ADMM proxy, but it cannot be directly deployed as a binary factor. Therefore, we absorb its effect into the existing intermediate scaling vector by solving

$$\boldsymbol{\gamma}^* = \arg\min_{\boldsymbol{\gamma}} \|\mathbf{A}_{\mathrm{target}} - \mathbf{A}_q \,\mathrm{Diag}(\boldsymbol{\gamma})\|_F^2, \tag{68}$$

where $\mathbf{A}_q$ is the SVID-projected discrete factor and $\boldsymbol{\gamma}$ is a column-wise correction vector along the intermediate dimension.

Let $\mathbf{a}_{t,k}$ and $\mathbf{a}_{q,k}$ denote the k-th columns of $\mathbf{A}_{\mathrm{target}}$ and $\mathbf{A}_q$, respectively. Since right multiplication by $\mathrm{Diag}(\boldsymbol{\gamma})$ independently rescales each column of $\mathbf{A}_q$, the objective in Eq. 68 can be decomposed into independent scalar problems:

$$\begin{aligned} &\|\mathbf{A}_{\mathrm{target}} - \mathbf{A}_q \,\mathrm{Diag}(\boldsymbol{\gamma})\|_F^2 \\ &= \sum_{k=1}^{r} \|\mathbf{a}_{t,k} - \gamma_k \mathbf{a}_{q,k}\|_2^2 . \end{aligned} \tag{69}$$

Thus, each γ_k can be optimized separately:

$$\gamma_k^* = \arg\min_{\gamma_k} \|\mathbf{a}_{t,k} - \gamma_k \mathbf{a}_{q,k}\|_2^2 . \tag{70}$$

Define the scalar objective

$$f_k(\gamma_k) = \|\mathbf{a}_{t,k} - \gamma_k \mathbf{a}_{q,k}\|_2^2 . \tag{71}$$

Expanding it gives

$$f_k(\gamma_k) = \mathbf{a}_{t,k}^\top \mathbf{a}_{t,k} - 2\gamma_k \mathbf{a}_{q,k}^\top \mathbf{a}_{t,k} + \gamma_k^2 \mathbf{a}_{q,k}^\top \mathbf{a}_{q,k}. \tag{72}$$

This is a quadratic function in γ_k. Its second derivative is

$$\frac{\partial^2 f_k}{\partial \gamma_k^2} = 2\mathbf{a}_{q,k}^\top \mathbf{a}_{q,k} = 2\,\|\mathbf{a}_{q,k}\|_2^2 \geq 0. \tag{73}$$

Therefore, $f_k(\gamma_k)$ is convex. Moreover, for the SVID-projected binary-scaled factor, $\mathbf{a}_{q,k}$ is nonzero in normal cases, so $\|\mathbf{a}_{q,k}\|_2^2 > 0$ and the objective is strictly convex. Hence, the minimizer exists and is unique.

Taking the first derivative and setting it to zero yields

$$\frac{\partial f_k}{\partial \gamma_k} = -2\mathbf{a}_{q,k}^\top \mathbf{a}_{t,k} + 2\gamma_k \mathbf{a}_{q,k}^\top \mathbf{a}_{q,k} = 0. \tag{74}$$

Solving for γ_k gives

$$\gamma_k^* = \frac{\mathbf{a}_{q,k}^\top \mathbf{a}_{t,k}}{\mathbf{a}_{q,k}^\top \mathbf{a}_{q,k}}. \tag{75}$$

Stacking all coordinates yields the vectorized closed-form solution

$$\boldsymbol{\gamma}^* = \frac{\operatorname{diag}\left(\mathbf{A}_q^\top \mathbf{A}_{\text{target}}\right)}{\operatorname{diag}\left(\mathbf{A}_q^\top \mathbf{A}_q\right)}. \tag{76}$$

In practice, a small constant ϵ can be added to the denominator for numerical stability:

$$\boldsymbol{\gamma}^* = \frac{\operatorname{diag}\left(\mathbf{A}_q^\top \mathbf{A}_{\text{target}}\right)}{\operatorname{diag}\left(\mathbf{A}_q^\top \mathbf{A}_q\right) + \epsilon}. \tag{77}$$

Finally, the obtained correction vector is folded into the intermediate scaling vector:

$$\mathbf{m}_1 \leftarrow \mathbf{m}_1 \odot \boldsymbol{\gamma}^*. \tag{78}$$

This operation keeps the binary sign matrix $\mathbf{A}_{\pm 1}$ unchanged and introduces no additional dense parameters. The compensated discrete factor remains within the double-factor binary parameterization, since

$$\begin{aligned}\mathbf{A}_q \operatorname{Diag}(\boldsymbol{\gamma}^*) &= \left(\mathbf{a} \odot \mathbf{A}_{\pm 1} \odot \mathbf{m}_1^\top\right) \operatorname{Diag}(\boldsymbol{\gamma}^*) \\ &= \mathbf{a} \odot \mathbf{A}_{\pm 1} \odot \left(\mathbf{m}_1 \odot \boldsymbol{\gamma}^*\right)^\top .\end{aligned} \tag{79}$$

Therefore, the continuous null-space-compensated target can be approximated by recalibrating only the existing scale vector, preserving the deployable binary structure.

The derivation for the right-factor update is symmetric. Given the compensated target $\mathbf{B}_{\text{target}}$ and the SVID-projected factor $\mathbf{B}_q$, we solve a row-wise scaling problem:

$$\boldsymbol{\eta}^* = \arg\min_{\boldsymbol{\eta}} \|\mathbf{B}_{\text{target}} - \operatorname{Diag}(\boldsymbol{\eta})\mathbf{B}_q\|_F^2 . \tag{80}$$

By the same least-squares argument, the closed-form solution is

$$\boldsymbol{\eta}^* = \frac{\operatorname{diag}\left(\mathbf{B}_{\text{target}}\mathbf{B}_q^\top\right)}{\operatorname{diag}\left(\mathbf{B}_q\mathbf{B}_q^\top\right)}, \tag{81}$$

or, with numerical stabilization,

$$\boldsymbol{\eta}^* = \frac{\operatorname{diag}\left(\mathbf{B}_{\text{target}}\mathbf{B}_q^\top\right)}{\operatorname{diag}\left(\mathbf{B}_q\mathbf{B}_q^\top\right) + \epsilon}. \tag{82}$$

This correction is folded into the corresponding intermediate scaling vector of the right factor.

A.2.6 Interaction-aware Property of Shapley Allocation

We explain why the Shapley-based structural-capacity game in HiSA can capture cross-layer interactions. The key is that Shapley values do not evaluate a layer in isolation. Instead, they measure the expected marginal effect of a layer under different coalition contexts.

Let $T = \{1, \ldots, L\}$ denote the set of Transformer layers, and let $v(S)$ be the value of a structural state where layers in S retain the original intermediate dimension while the remaining layers are degraded to the proxy dimension. For layer l, its Shapley value is

$$\phi_l = \sum_{S \subseteq T \setminus \{l\}} \frac{|S|!(L-|S|-1)!}{L!} \left[v(S \cup \{l\}) - v(S)\right] \tag{83}$$

The marginal term $v(S \cup \{l\}) - v(S)$ evaluates the contribution of layer l under a specific context S. Different subsets S correspond to different compression states of the remaining layers. Therefore, the same layer is evaluated under many upstream

and downstream compression conditions, rather than under a fixed isolated setting.

If compression errors were independent and additive across layers, the marginal effect of layer l would be context-invariant:

$$v(S \cup \{l\}) - v(S) = \Delta_l, \quad \forall S \subseteq T \setminus \{l\}. \quad (84)$$

In this special case, isolated local sensitivity would be sufficient, and the Shapley value would reduce to the same contribution Δ_l.

However, deep Transformers generally violate this additive assumption. Compressing one layer changes the hidden-state distribution received by subsequent layers, and downstream layers may amplify, suppress, or compensate the earlier error. As a result, the marginal effect of layer l depends on the coalition context:

$$v(S \cup \{l\}) - v(S) \neq v(S' \cup \{l\}) - v(S'), \quad S \neq S'. \quad (85)$$

This context dependence is exactly the cross-layer interaction that isolated local reconstruction error cannot capture.

The permutation view of Shapley values makes this interaction-aware property clearer. For a random permutation π of layers, let $\mathrm{Pre}_\pi(l)$ denote the set of layers appearing before l in π. Then,

$$\phi_l = \mathbb{E}_\pi \left[v(\mathrm{Pre}_\pi(l) \cup \{l\}) - v(\mathrm{Pre}_\pi(l)) \right]. \quad (86)$$

Thus, each sampled permutation evaluates layer l under a different structural context.

In HiSA, we approximate this expectation by progressively degrading layers along sampled permutations and measuring the induced increase in end-to-end NLL:

$$\Delta v_l^{(m)} = v_{\mathrm{NLL}}(S_l^{(m)} \setminus \{l\}) - v_{\mathrm{NLL}}(S_l^{(m)}), \quad (87)$$

where $S_l^{(m)}$ denotes the set of layers that still retain the original dimension before layer l is degraded in the m-th permutation. Averaging over M sampled permutations gives

$$\Phi_l = \frac{1}{M} \sum_{m=1}^{M} \Delta v_l^{(m)}. \quad (88)$$

This estimator differs from isolated local sensitivity in two aspects. First, the value function is the full-model NLL, so the measured effect includes downstream propagation and prediction-level changes. Second, each marginal contribution is evaluated under a different coalition of compressed and uncompressed layers, so the estimate reflects how layer l interacts with other layers under different structural states. Therefore, the layer-level Shapley score in HiSA is interaction-aware and is more suitable for allocating global structural budgets than uniform dimensions or isolated reconstruction errors.

A.2.7 Dual Distortion Decomposition for Block-Level Allocation

We provide the motivation for using both directional and magnitude distortions in block-level HiSA. The key observation is that a single MSE score entangles two different types of representation errors: angular drift and activation-energy shift. Let $\mathbf{h}$ and $\hat{\mathbf{h}}$ denote the reference and compressed hidden states of a token, respectively. Writing

$$\mathbf{h} = \|\mathbf{h}\|_2 \mathbf{u}, \qquad \hat{\mathbf{h}} = \|\hat{\mathbf{h}}\|_2 \hat{\mathbf{u}}, \quad (89)$$

where $\|\mathbf{u}\|_2 = \|\hat{\mathbf{u}}\|_2 = 1$, the squared error can be decomposed as

$$\begin{aligned} \|\mathbf{h} - \hat{\mathbf{h}}\|_2^2 &= \|\mathbf{h}\|_2^2 + \|\hat{\mathbf{h}}\|_2^2 - 2\langle \mathbf{h}, \hat{\mathbf{h}} \rangle \\ &= \left(\|\mathbf{h}\|_2 - \|\hat{\mathbf{h}}\|_2 \right)^2 \\ &\quad + 2\|\mathbf{h}\|_2 \|\hat{\mathbf{h}}\|_2 \left(1 - \langle \mathbf{u}, \hat{\mathbf{u}} \rangle\right). \end{aligned} \quad (90)$$

The first term measures the mismatch in representation magnitude, while the second term measures the angular deviation between the two normalized directions. Therefore, MSE is a coupled measure of magnitude and directional errors, and a single scalar may hide which failure mode dominates.

This distinction is important for Transformer blocks. In self-attention, the attention logits are computed from query–key similarities:

$$\mathbf{S} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}. \quad (91)$$

For two token vectors $\mathbf{q}$ and $\mathbf{k}$,

$$\mathbf{q}^\top \mathbf{k} = \|\mathbf{q}\|_2 \|\mathbf{k}\|_2 \cos(\theta_{\mathbf{q},\mathbf{k}}). \quad (92)$$

Thus, angular changes in query or key representations can directly alter token-matching patterns and attention distributions, even when vector norms are not severely changed. This makes Q/K projections particularly sensitive to directional distortion.

By contrast, value projections, output projections, and MLP branches mainly inject transformed activations into the residual stream. Their compression errors often manifest as changes in activation

energy, channel scale, or residual update strength. For a residual update

$$\mathbf{h}_{l+1} = \mathbf{h}_l + F_l(\mathbf{h}_l), \tag{93}$$

a magnitude mismatch in the compressed branch $\hat{F}_l$ changes the scale of the injected update:

$$\|\hat{F}_l(\mathbf{h}_l)\|_2 \neq \|F_l(\mathbf{h}_l)\|_2. \tag{94}$$

Such scale shifts can accumulate across layers and disturb downstream normalization, residual balance, and activation dynamic range. This makes magnitude distortion an important signal for V/O projections and MLP modules.

However, these module tendencies should not be hard-coded as fixed priors, since their sensitivity can vary across layers, model families, and compression budgets. HiSA therefore measures both distortions directly from layer outputs. The directional error

$$r_{\text{dir}} = \mathbb{E}_{b,t}\left[1 - \frac{\langle h_{b,t}^{\text{ref}}, \hat{h}_{b,t}\rangle}{\|h_{b,t}^{\text{ref}}\|_2 \|\hat{h}_{b,t}\|_2 + \epsilon}\right] \tag{95}$$

captures angular drift of token representations, while the magnitude error

$$r_{\text{mag}} = \mathbb{E}_{b,t}\left[\frac{\left|\|\hat{h}_{b,t}\|_2 - \|h_{b,t}^{\text{ref}}\|_2\right|}{\|h_{b,t}^{\text{ref}}\|_2 + \epsilon}\right] \tag{96}$$

captures relative activation-energy shift. By estimating Shapley marginal contributions separately under these two metrics, HiSA identifies whether a submodule is vulnerable through directional mismatch, magnitude mismatch, or both. The subsequent Soft-OR fusion ensures that a module with strong sensitivity in either failure mode can still receive sufficient structural capacity.

A.2.8 Motivation for Size-Aware Sensitivity Density

We explain why HiSA converts the fused vulnerability score into a size-aware sensitivity density before allocating the module-level budget. For brevity, let v_i denote $v_{\text{base}}^{(i)}$, and let $|\mathbf{W}_i|$ denote the parameter count of submodule i. If the allocation directly uses v_i as the module weight, then the total budget received by a module is affected by both its vulnerability and its size. Specifically, under a layer budget B_l, a generic allocation weight ω_i gives

$$b_i = B_l \cdot \frac{P_l \omega_i}{\sum_{j\in\mathcal{N}_l} |\mathbf{W}_j|\omega_j}, \qquad P_l = \sum_{j\in\mathcal{N}_l} |\mathbf{W}_j|. \tag{97}$$

The actual number of allocated bits for module i is therefore

$$|\mathbf{W}_i| b_i = B_l P_l \cdot \frac{|\mathbf{W}_i|\omega_i}{\sum_{j\in\mathcal{N}_l} |\mathbf{W}_j|\omega_j}. \tag{98}$$

If we set $\omega_i = v_i$, then the total allocated bits become proportional to $|\mathbf{W}_i| v_i$. As a result, a large module can dominate the layer budget even when its vulnerability per parameter is not high.

To reduce this size-induced bias, HiSA first converts vulnerability into a sensitivity density:

$$d_i = v_i / |\mathbf{W}_i|. \tag{99}$$

This quantity measures vulnerability per parameter, rather than the aggregate vulnerability of the whole module. However, using density alone may over-emphasize very small modules, whose Shapley estimates can be noisier due to fewer parameters and smaller output contribution. Therefore, HiSA applies a smoothing exponent:

$$\omega_i = (d_i + \epsilon)^\alpha. \tag{100}$$

Substituting this into the total allocated bits gives

$$|\mathbf{W}_i| b_i \propto |\mathbf{W}_i| \left(v_i / |\mathbf{W}_i| + \epsilon\right)^\alpha. \tag{101}$$

Ignoring the small ϵ for clarity, this becomes

$$|\mathbf{W}_i| b_i \propto |\mathbf{W}_i|^{1-\alpha} v_i^\alpha. \tag{102}$$

Thus, α controls the trade-off between size-proportional allocation and vulnerability-density allocation. When $\alpha = 0$, all modules receive the same average bit-width and total allocated bits are proportional to module size. When $\alpha = 1$, the total allocated bits are mainly determined by the fused vulnerability v_i, reducing the dominance of large modules. Intermediate values of α interpolate between these two extremes.

As an illustrative example, consider a Qwen3-8B-style decoder layer. With hidden size 4096 and intermediate size 22016, an attention projection such as $\mathbf{W}_q$ has approximately

$$4096 \times 4096 \approx 16.8\text{M} \tag{103}$$

parameters, while an MLP projection such as $\mathbf{W}_{\text{up}}$ has approximately

$$4096 \times 22016 \approx 90.2\text{M} \tag{104}$$

parameters. The MLP projection is therefore about $5.4\times$ larger than the attention projection. Suppose the fused vulnerabilities are $v_{\text{mlp}} = 0.6$ and

$v_{\text{attn}} = 0.3$. If raw vulnerability is used as the allocation weight, the ratio of allocated bits is

$$\frac{|\mathbf{W}_{\text{mlp}}|v_{\text{mlp}}}{|\mathbf{W}_{\text{attn}}|v_{\text{attn}}} \approx 5.4 \times 2 = 10.8. \tag{105}$$

Thus, the larger MLP module receives more than ten times the total budget, even though its vulnerability score is only twice as large. By contrast, with density smoothing and $\alpha = 1$, the size factor is approximately canceled in the total-bit allocation:

$$\frac{|\mathbf{W}_{\text{mlp}}|d_{\text{mlp}}}{|\mathbf{W}_{\text{attn}}|d_{\text{attn}}} = \frac{v_{\text{mlp}}}{v_{\text{attn}}} = 2. \tag{106}$$

With $0 < \alpha < 1$, HiSA obtains a smoother interpolation, reducing large-module dominance while avoiding excessive sensitivity to noisy estimates from small modules. This motivates the size-aware density and smoothing design used in Eq. 19, where P_l and ω_i are expanded inline in the main text for compactness.

A.3 Effective Bit-Width of Binary PTQ Methods

Although binary PTQ methods represent most quantized weights with 1-bit signs, their deployable storage is not determined by binary weights alone. To reconstruct the quantized matrix, one also needs to store residual binary components, BF16 scaling factors and means, salient-column indicators, group bitmaps, and, for structured sparsity, sparsity indices. Therefore, we report the effective bits per weight (BPW) by counting all bits required for weight reconstruction.

For a weight matrix $W \in \mathbb{R}^{n\times m}$, let n and m denote the number of rows and columns, respectively. Let c be the number of salient columns selected in W, k be the block size, and $q = \lceil m/k \rceil$ be the number of column blocks per row. Following common implementations, all reconstruction coefficients, including scaling factors and means, are stored in BF16. If the quantized representation requires M bits in total, then

$$\text{BPW} = \frac{M}{nm}. \tag{107}$$

BiLLM (Huang et al., 2024a). BiLLM separates the weight matrix into salient and non-salient columns. For the salient columns, BiLLM adopts second-order residual binarization, which stores two binary tensors for the original and residual components. Thus, the binary storage of the salient part is $2nc$. In addition, each column block stores three BF16 row-wise reconstruction coefficients, corresponding to α_1, α_2, and the merged mean μ, which costs $3 \times 16nq = 48nq$ bits. For the non-salient columns, BiLLM uses first-order binarization with two distribution groups, which stores $n(m-c)$ binary signs and two sets of BF16 scale/mean coefficients, costing $2 \times 2 \times 16nq = 64nq$ bits. Finally, BiLLM stores a group bitmap over the whole matrix and a salient-column bitmap, which cost nm and m bits, respectively. Therefore,

$$\begin{aligned} M_{\text{BiLLM}} &= (2nc + 48nq) + \big(n(m-c) + 64nq\big) \\ &\quad + nm + m = 2nm + nc + m + 112nq. \end{aligned} \tag{108}$$

Dividing Eq. 108 by nm gives

$$\text{BPW}_{\text{BiLLM}} = 2 + \frac{c}{m} + \frac{112q}{m} + \frac{1}{n}. \tag{109}$$

STBLLM$_{4:8}$ (Dong et al., 2024). STBLLM further introduces structured $N:M$ sparsity on top of BiLLM-style binary PTQ. We only consider the 4:8 setting used in our comparison. The salient part still follows residual binarization, so it costs $2nc + 48nq$ bits, as in BiLLM. For the non-salient part, 4:8 sparsity retains only half of the binary weights. Therefore, the retained binary signs contribute $\frac{1}{2}n(m-c)$ bits. STBLLM also divides non-salient weights into sparse, intermediate, and dense regions using trisection search, and each retained value needs a 2-bit group identifier; this contributes $\frac{1}{2} \cdot 2nm$ bits. Moreover, each 4:8 block needs to store which four positions are retained among eight positions. Since $\lceil \log_2 \binom{8}{4} \rceil = 7$, the sparsity-index overhead is $\frac{7}{8}n(m-c)$ bits. The three non-salient groups require three sets of BF16 scale/mean coefficients, costing $3 \times 2 \times 16nq = 96nq$ bits. Together with the salient-column bitmap, the total storage is

$$\begin{aligned} M_{\text{STBLLM}_{4:8}} &= (2nc + 48nq) + \frac{1}{2}\big[n(m-c) + 2nm\big] \\ &\quad + \frac{7}{8}n(m-c) + 96nq + m \\ &= \frac{19}{8}nm + \frac{5}{8}nc + 144nq + m. \end{aligned} \tag{110}$$

Thus, the effective BPW is

$$\text{BPW}_{\text{STBLLM}_{4:8}} = \frac{19}{8} + \frac{5c}{8m} + \frac{144q}{m} + \frac{1}{n}. \tag{111}$$

ARB-LLM$_{\text{RC}}$ (Li et al., 2024b). ARB-LLM$_{\text{RC}}$ improves BiLLM by alternatingly refining binarization parameters and introducing row-column-wise

scaling to better preserve column-wise deviations. For salient columns, it stores two binary residual components, giving $2nc$ bits. Different from purely row-wise BiLLM, ARB-LLM$_{\mathrm{RC}}$ stores both row-wise and column-wise BF16 coefficients for the residual representation. Specifically, the salient part requires $(2nq + 2c) \times 16$ bits, where $2nq$ corresponds to row-wise coefficients over two residual components and $2c$ corresponds to column-wise coefficients for the salient columns. For the non-salient part, ARB-LLM$_{\mathrm{RC}}$ stores $n(m-c)$ binary signs and two groups of row-column BF16 coefficients, yielding $(nq + m - c) \times 16 \times 2$ bits. It also requires a group bitmap and a salient-column bitmap, costing nm and m bits. Therefore,

$$\begin{aligned} M_{\mathrm{ARB_{RC}}} &= \big(2nc + 16(2nq + 2c)\big) \\ &\quad + \big(n(m-c) + 32(nq + m - c)\big) \\ &\quad + nm + m = 2nm + nc + 33m + 64nq. \end{aligned} \tag{112}$$

The corresponding BPW is

$$\mathrm{BPW}_{\mathrm{ARB_{RC}}} = 2 + \frac{c}{m} + \frac{64q}{m} + \frac{33}{n}. \tag{113}$$

HBLLM$_{\mathrm{row}}$ (Chen et al., 2025). HBLLM$_{\mathrm{row}}$ extends BiLLM with Haar-domain row-wise quantization and frequency-aware grouping. In the non-salient branch, HBLLM$_{\mathrm{row}}$ uses FillAvg to fill salient positions before applying row-wise Haar quantization. As a result, the non-salient Haar branch stores a dense binary representation over nm positions rather than only $n(m-c)$ positions. Its BF16 coefficient overhead is $3 \times 16nq \times 2 = 96nq$ bits because the row-wise Haar branch uses two frequency-aware groups with three BF16 coefficients per group. For the salient branch, HBLLM$_{\mathrm{row}}$ stores nc binary signs and two groups of BF16 coefficients, costing $2 \times 2 \times 16nq = 64nq$ bits. In addition, the FillAvg-based row branch requires a group bitmap covering both the original matrix and the filled salient positions, giving $n(m+c)$ bits, while the salient-column bitmap costs m bits. Hence,

$$\begin{aligned} M_{\mathrm{HBLLM_{row}}} &= (nm + 96nq) + (nc + 64nq) \\ &\quad + n(m+c) + m \\ &= 2nm + 2nc + m + 160nq. \end{aligned} \tag{114}$$

Therefore,

$$\mathrm{BPW}_{\mathrm{HBLLM_{row}}} = 2 + \frac{2c}{m} + \frac{160q}{m} + \frac{1}{n}. \tag{115}$$

AF1 (ours). For a target linear matrix $W_i \in \mathbb{R}^{d_{\mathrm{out},i} \times d_{\mathrm{in},i}}$, AF1 stores two binary sign factors $A_i \in \{-1,+1\}^{d_{\mathrm{out},i} \times r_i}$ and $B_i \in \{-1,+1\}^{r_i \times d_{\mathrm{in},i}}$, together with three BF16 scale vectors $a_i \in \mathbb{R}^{d_{\mathrm{out},i}}$, $m_i \in \mathbb{R}^{r_i}$, and $b_i \in \mathbb{R}^{d_{\mathrm{in},i}}$. Therefore, the deployable storage of this matrix is

$$M_i^{\mathrm{AF1}} = r_i(d_{\mathrm{out},i} + d_{\mathrm{in},i}) + 16(d_{\mathrm{out},i} + r_i + d_{\mathrm{in},i}). \tag{116}$$

The corresponding model-level effective BPW is

$$\mathrm{BPW}_{\mathrm{AF1}} = \frac{\sum_i M_i^{\mathrm{AF1}}}{\sum_i d_{\mathrm{out},i} d_{\mathrm{in},i}}. \tag{117}$$

In practice, HiSA assigns the module-wise budgets and AF1 chooses the intermediate dimensions $\{r_i\}$ under the model-level constraint in Eq. 117. The calibration statistics, ADMM variables, null-space projectors, and teacher logits are used only during quantization and are not stored for deployment. Thus, unlike prior binary PTQ baselines, AF1 does not require salient-column masks, group bitmaps, residual binary tensors, or structured-sparsity indices at inference time.

These derivations show that existing binary PTQ baselines are not purely 1-bit in effective storage. Their BPW is increased by second-order residual binary tensors, BF16 reconstruction coefficients, saliency masks, group bitmaps, and structured-sparsity metadata. In contrast, AF1 explicitly controls the deployable storage through its binary-factor dimensions and lightweight scale vectors, explaining why the nominal binary width and the actual deployable BPW can differ substantially for prior methods but remain budget-controlled in AF1.

A.4 More Experimental Results

A.4.1 More Detailed Results

In this appendix, we provide the full expanded results corresponding to Table 1 in the main paper. Table 4 reports the task-level zero-shot accuracy on ARC-Easy, ARC-Challenge, HellaSwag, LAMBADA-openai, LAMBADA-standard, PIQA, and WinoGrande for each evaluated model. The expanded breakdown shows that the advantage of AF1 is not caused by isolated improvements on a single benchmark, but is consistently reflected across different task categories, including commonsense reasoning, language modeling, and cloze-style completion. Under the 2.50-bit setting, AF1 achieves the best averaged zero-shot accuracy for all evaluated model families and obtains the

strongest or near-strongest performance on most individual tasks, while also yielding the lowest WikiText2 perplexity in Table 1. Notably, even at the strict 1.00-bit setting, AF1 remains competitive with or superior to several higher-bit baselines, demonstrating that the proposed binary factorization and hierarchical budget allocation preserve task-level generalization under an extremely constrained weight budget. These detailed results further support the main conclusion that AF1 improves both average performance and per-task robustness over existing binary PTQ methods.

A.4.2 Ablation on the Residual-Energy Threshold

We further study the sensitivity of NABF to the residual-energy threshold η used in the adaptive null-space cutoff. This threshold controls the size of the selected trailing subspace: a smaller η yields a more conservative null-space estimate, while a larger η introduces more low-energy directions for compensation. Table 5 reports WikiText2 perplexity on LLaMA-3-8B, Qwen3-14B, and Gemma-3-4B-it when sweeping η from 0.01 to 0.09. Overall, $\eta = 0.01$ achieves the lowest average perplexity and is therefore used as the default setting.

A.4.3 Ablation on Allocation-Sensitivity Metrics

We further investigate how different sensitivity metrics affect structural budget allocation under the same average weight-bit budget. Specifically, we compare five representative allocation metrics and three HiSA variants on LLaMA-3-8B and Qwen3-8B, while keeping the binary factorization procedure, calibration data, and reconstruction settings unchanged. The quantization quality is evaluated by WikiText2 perplexity (PPL; lower is better).

- **Uniform** assigns the same structural budget to all layers and submodules, ignoring the non-uniform sensitivity of different components.
- **LIM** (Layer Input Modification) measures layer importance by the cosine drift between the layer input and output representations, where larger input–output changes indicate higher sensitivity.
- **ZD** (Z-score Distribution) estimates sensitivity by the fraction of statistically abnormal weights in each layer, assuming that layers with more outlier weights are harder to binarize.
- **Activation-based scoring** uses the Frobenius norm of layer activations as the allocation criterion, based on the assumption that layers with larger activation energy carry more important information.
- **NLL Increase** measures the validation negative log-likelihood increase caused by quantizing one layer at a time, while keeping the remaining layers at a higher-precision reference.
- **Only Directional** keeps the hierarchical Shapley allocation framework but uses only the directional distortion for module-level budget allocation.
- **Only Magnitude** keeps the hierarchical Shapley allocation framework but uses only the magnitude distortion for module-level budget allocation.
- **HiSA** denotes the full dual-distortion allocation strategy, which fuses directional and magnitude sensitivities with the Soft-OR rule.

As shown in Table 6, HiSA consistently achieves the lowest WikiText2 perplexity on both models under the same 1.00-bit weight budget. Compared with uniform allocation, HiSA reduces perplexity from 35.84 to 22.18 on LLaMA-3-8B and from 28.92 to 16.26 on Qwen3-8B, indicating that uniformly assigning the same structural capacity to all layers is highly suboptimal for genuine 1-bit PTQ. Local sensitivity metrics such as LIM, ZD, and activation-based scoring improve over the uniform baseline in most cases, but their performance remains clearly worse than HiSA because they rely on isolated representation shifts, weight outlier statistics, or activation magnitudes, and therefore cannot fully reflect the end-to-end effect of quantization errors. NLL Increase provides a stronger baseline by directly measuring the loss perturbation caused by single-layer quantization, and it achieves the best result among these non-Shapley allocation baselines. Nevertheless, HiSA further reduces PPL from 25.86 to 22.18 on LLaMA-3-8B and from 19.37 to 16.26 on Qwen3-8B. This improvement shows that evaluating layer sensitivity in isolation is still insufficient, since quantization errors can accumulate, propagate, and be partially compensated across layers.

We further isolate the contribution of the dual-distortion design by comparing full HiSA with two single-distortion variants. Only Directional uses the directional Shapley score alone, while Only Magnitude uses the magnitude Shapley score alone. Both variants underperform the Soft-OR fusion strategy, showing that the two distortion modes capture complementary failure patterns. Specifically, on LLAMA-3-8B, HiSA reduces PPL from 24.59 and 23.33 to 22.18 compared with Only Directional and Only Magnitude, respectively. On QWEN3-8B, HiSA similarly reduces PPL from 18.15 and 17.59 to 16.26. These results confirm that the strong 1-bit performance of AF1 comes not only from hierarchical Shapley allocation, but also from explicitly preserving modules that are vulnerable in either directional or magnitude distortion.

A.4.4 Additional Ablation on HiSA Hyperparameters

We provide additional ablation results for HiSA on LLAMA-3-8B and GEMMA-3-4B-IT to further verify the robustness of the default hyperparameter configuration. Together with the Qwen3-8B results in Fig. 3, these additional curves show that the behavior of HiSA is consistent across different model families and scales. We analyze four key hyperparameters: the layer-level temperature coefficient τ, the layer-level sampling number M_{layer}, the block-level smoothing exponent α, and the block-level sampling number M_{block}.

For the layer-level allocation, τ controls how concentrated the structural budget is across layers. When τ is too small, the allocation becomes overly sharp and assigns excessive capacity to a small number of layers, which leaves many other layers under-allocated and leads to a large increase in perplexity. When τ becomes too large, the allocation gradually approaches a uniform distribution and weakens the effect of layer-wise sensitivity estimation. As shown in Fig. 6 and Fig. 7, both models achieve strong performance around $\tau = 1.0$, which is consistent with the trend observed on Qwen3-8B. This indicates that a moderately concentrated layer-wise allocation is preferable: it preserves the benefit of sensitivity-aware budget assignment while avoiding excessive concentration on only a few layers.

The sampling number M_{layer} determines the accuracy of the Monte Carlo Shapley estimation at the layer level. When only a small number of permutations is used, the estimated layer contribution is noisy, and the resulting allocation is unstable. Increasing M_{layer} consistently reduces perplexity on both LLAMA-3-8B and GEMMA-3-4B-IT, and the improvement gradually saturates after a moderate number of samples. This trend confirms that the layer-level Shapley score benefits from sufficient permutation sampling, but it does not require an excessively large sampling budget to obtain a reliable allocation.

For the block-level allocation, α controls the strength of sensitivity-density redistribution among submodules inside each layer. A small-to-moderate value of α leads to stable performance, because it allows sensitive submodules to receive more capacity while keeping the allocation sufficiently smooth. In contrast, a large α over-amplifies the difference between submodule sensitivities and causes the budget to be concentrated on a small subset of modules. This behavior substantially degrades perplexity on both models, especially when α is larger than 0.4. Therefore, the default setting $\alpha = 0.2$ provides a robust balance between sensitivity awareness and allocation smoothness.

The block-level sampling number M_{block} shows a similar pattern to M_{layer}. Using too few samples results in unreliable submodule-level contribution estimates and noticeably worse perplexity. As M_{block} increases, the estimated block-level importance becomes more stable, and the perplexity quickly decreases before reaching a near-saturated region. Although using more samples can sometimes bring marginal additional improvement, the gain becomes much smaller after the estimation stabilizes. Therefore, we adopt $M_{\text{layer}}/M_{\text{block}} = 60/40$ as the default configuration, which provides a good trade-off between quantization quality and allocation overhead.

Overall, these additional ablations support the hyperparameter choices used in the main experiments. The trends on LLAMA-3-8B, QWEN3-8B, and GEMMA-3-4B-IT consistently show that HiSA is not sensitive to a narrow, model-specific setting. Instead, it maintains stable performance under a reasonable range of hyperparameters, while clearly degrading only under extreme allocation regimes such as overly concentrated layer budgets or overly aggressive block-level sensitivity amplification. These results further demonstrate that the effectiveness of HiSA comes from its hierarchical sensitivity modeling rather than from delicate hyperparameter tuning.

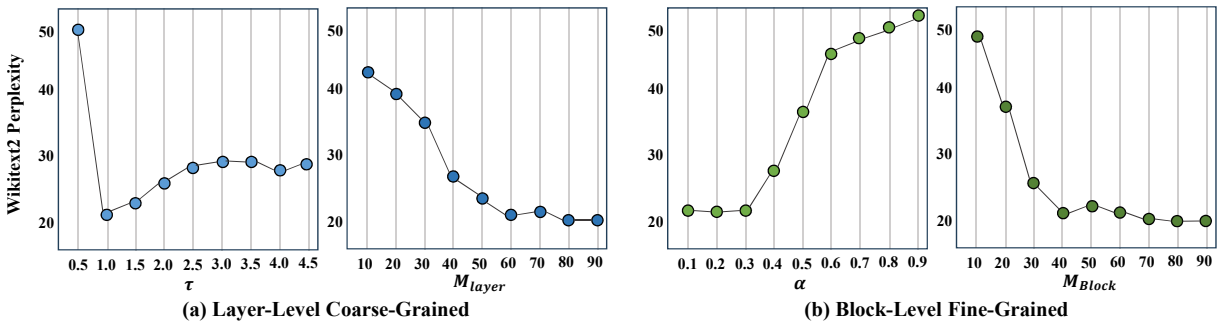


Figure 6: Additional hyperparameter sensitivity of HiSA on LLAMA-3-8B. We report WikiText2 perplexity under different values of the layer-level temperature coefficient τ, the layer-level sampling number M_{layer}, the block-level smoothing exponent α, and the block-level sampling number M_{block}.

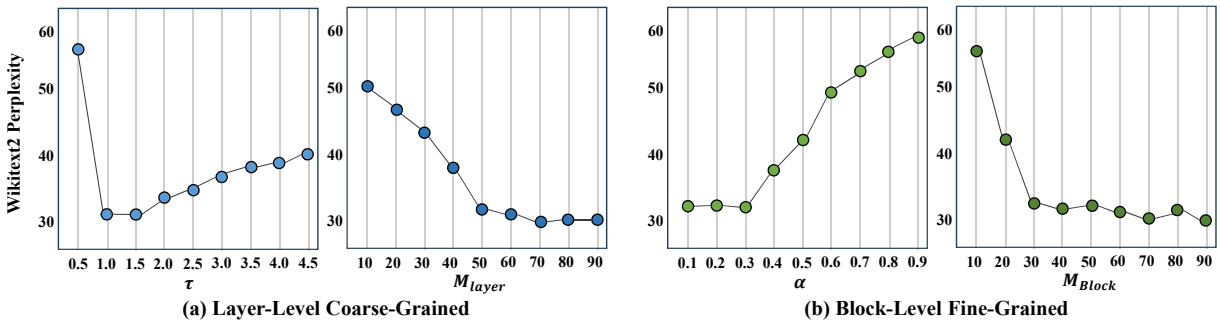


Figure 7: Additional hyperparameter sensitivity of HiSA on GEMMA-3-4B-IT. We report WikiText2 perplexity under different values of the layer-level temperature coefficient τ, the layer-level sampling number M_{layer}, the block-level smoothing exponent α, and the block-level sampling number M_{block}.

A.4.5 Ablation on Calibration Data

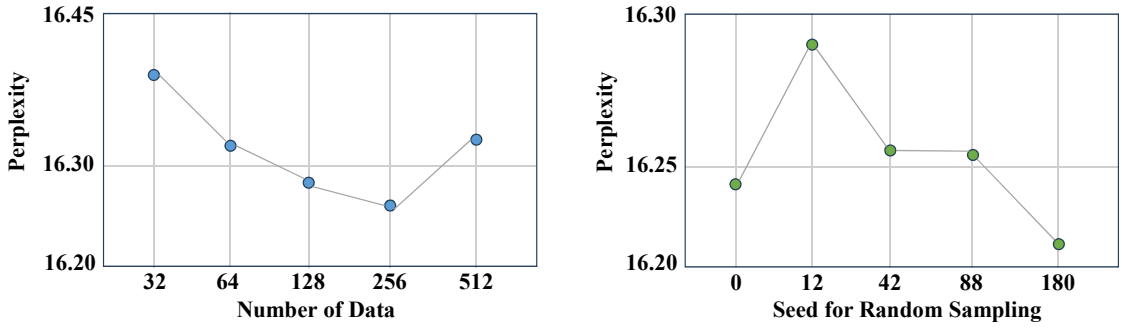
We further investigate the influence of calibration data on AF1. Specifically, we use C4 as the calibration corpus and evaluate the quantized model by WikiText2 perplexity, while keeping all quantization hyperparameters and reconstruction settings unchanged. We consider two factors: the number of calibration samples and the random seed used for calibration-data sampling. For the calibration-size study, we vary the number of C4 samples from 32 to 512. For the seed-sensitivity study, we fix the calibration size and change the random seed, which affects both sample selection and token order.

As shown in Fig. 8, AF1 exhibits stable performance under different calibration-data configurations. When the number of calibration samples increases from 32 to 512, the WikiText2 perplexity remains within a narrow range, with the best result obtained around 256 samples and only a slight degradation when using more samples. This indicates that AF1 does not require a large calibration set to obtain reliable reconstruction statistics. Similarly, changing the random seed leads to only minor fluctuations in perplexity, and all tested seeds produce comparable results. These observations suggest that AF1 is insensitive to both calibration-set size and calibration-sampling randomness. Therefore, a small representative subset of C4 is sufficient for stable post-training binarization, which reduces calibration overhead and improves the practicality of AF1 in deployment scenarios.

A.5 Dialog Examples

To further illustrate the qualitative impact of different binarization schemes, Table 7 presents representative generations from Qwen3-14B-Instruct under full precision and different binary PTQ methods, including BiLLM, ARB-LLM, HBLLM, and our proposed AF1. Different from open-ended dialogue prompts, we use short questions with objective answers, covering factual recall, elementary arithmetic, and basic commonsense knowledge. This design allows us to directly examine whether the compressed models preserve the key answer content and whether their generations remain stable under extreme weight compression.

In addition to checking the ground-truth answers, we further use GPT-5.5 as an external LLM judge to provide a consistent qualitative assessment of each response. Specifically, the judge is asked to evalu-



(a) Change of Number

(b) Change of Seed

Figure 8: Ablation on calibration data. We use C4 as the calibration corpus and report WikiText2 perplexity after quantization. (a) varying the number of calibration samples. (b) varying the random seed for calibration-data sampling.

ate whether a generation contains the correct key answer, whether it remains semantically coherent, and whether it suffers from repetition, symbol collapse, or irrelevant continuation. The LLM judge is used only for qualitative annotation rather than as a replacement for standard benchmark evaluation, and its judgments are reported in the assessment column of Table 7. This provides a more systematic way to distinguish factually correct but unstable outputs from truly coherent generations.

As shown in Table 7, existing binary PTQ methods exhibit different types of generation degradation. BiLLM frequently collapses into repeated tokens, symbols, or self-reinforcing phrases, and in some cases produces factually incorrect answers despite the simplicity of the prompts. ARB-LLM preserves more factual information than BiLLM on several questions, but it can still produce redundant or unstable continuations, especially when the generation length increases. HBLLM achieves relatively stable generations in these examples, but this comes at a substantially larger storage cost, with an effective weight budget exceeding 3 BPW in our setting. Therefore, its qualitative advantage should be interpreted together with its much higher bit-width rather than as evidence of genuine 1-bit compression.

In contrast, AF1 targets a genuinely low-bit regime, using only around 1 BPW for weights, while prior binary PTQ baselines typically require approximately 2.5–3.5 BPW due to additional full-precision scaling factors, grouping metadata, or structured compensation terms. Despite this much stricter storage budget, AF1 preserves the key answer content across most selected examples and avoids the severe symbolic collapse observed in prior binary PTQ methods. For instance, AF1 correctly answers the arithmetic, capital-city, average-speed, red-planet, and triangle-angle questions while maintaining readable sentence structure. Although long-form repetitions may still occur in some cases, these qualitative results indicate that AF1 achieves a more favorable trade-off between storage efficiency and generation quality, demonstrating that near-1-BPW LLM compression can retain basic language generation ability under extreme quantization.

A.6 Use of Large Language Models

In preparing this manuscript, we employed large language models (LLMs) as writing assistants. Specifically, LLMs were used for grammar correction, wording improvement, and stylistic polishing of the text. In certain cases, LLMs were also leveraged to rephrase or restructure preliminary drafts of specific sections to improve clarity and readability. Importantly, all conceptual contributions, technical methods, experimental designs, and analyses were conceived and developed by the authors without reliance on LLMs. The final responsibility for the accuracy and integrity of the content rests entirely with the authors.

Model	Method	#Bits(W)	AE↑	AC↑	HS↑	LO↑	LS↑	PQ↑	WG↑	Avg.[7] ↑
LLaMA-3-8B	BF16	16	77.90	52.82	79.07	75.63	68.58	80.63	72.93	72.51
	RTN	1.00	24.71	25.94	26.63	0.00	0.00	51.85	51.62	25.82
	BiLLM	2.88	36.78	22.18	34.81	28.76	13.00	58.49	53.67	35.38
	STBLLM$_{(4:8)}$	3.50	31.10	21.76	31.29	11.84	5.88	54.52	49.64	29.43
	ARB-LLM$_{RC}$	2.51	48.78	28.33	43.55	50.65	29.83	65.61	56.75	46.21
	HBLLM$_{row}$	3.25	54.12	32.68	60.22	58.51	46.40	70.35	65.19	55.35
	AF1	2.50	**72.43**	**44.62**	**73.81**	**70.56**	**62.51**	**78.29**	**69.22**	**67.35**
	AF1	1.00	44.51	26.49	42.01	47.35	36.48	65.00	56.54	45.48
LLaMA-3-70B	BF16	16	86.07	64.33	84.92	79.31	73.96	84.60	80.35	79.08
	RTN	1.00	25.31	24.33	25.59	0.00	0.00	50.61	48.29	24.88
	BiLLM	2.88	25.34	20.73	33.18	9.41	5.44	52.39	51.78	28.32
	STBLLM$_{(4:8)}$	3.50	25.67	25.94	27.23	1.84	4.29	51.03	49.88	26.55
	ARB-LLM$_{RC}$	2.51	61.71	42.48	67.44	65.68	53.54	69.97	64.53	60.76
	HBLLM$_{row}$	3.25	24.83	27.05	74.70	70.66	50.96	50.60	73.56	53.19
	AF1	2.50	**71.89**	**48.04**	**83.43**	**76.93**	**70.74**	**77.20**	**77.66**	**72.27**
	AF1	1.00	55.56	36.69	73.92	68.60	61.36	68.39	68.11	61.80
Qwen3-8B	BF16	16	80.93	56.74	74.98	64.18	61.11	77.37	68.35	69.09
	RTN	1.00	25.84	25.17	26.56	0.00	0.00	49.95	49.41	25.28
	BiLLM	2.88	34.97	23.81	37.69	18.75	12.71	56.80	51.14	33.70
	STBLLM$_{(4:8)}$	3.50	31.78	22.61	29.89	5.72	2.99	53.81	49.33	28.02
	ARB-LLM$_{RC}$	2.51	59.09	36.35	49.84	41.86	34.29	68.72	58.80	49.85
	HBLLM$_{row}$	3.25	58.38	36.18	60.51	50.32	44.91	71.82	61.17	54.76
	AF1	2.50	**78.16**	**55.03**	**72.91**	**61.42**	**58.98**	**77.31**	**71.82**	**67.95**
	AF1	1.00	62.07	37.91	55.76	50.95	45.24	70.87	60.97	54.82
Qwen3-14B	BF16	16	83.08	60.49	78.82	67.84	64.47	79.76	72.85	72.47
	RTN	1.00	24.62	26.88	26.14	0.00	0.00	51.41	49.80	25.55
	BiLLM	2.88	56.57	34.56	55.70	48.67	43.90	70.62	64.72	53.53
	STBLLM$_{(4:8)}$	3.50	47.52	27.47	44.33	24.72	22.86	63.06	56.20	40.88
	ARB-LLM$_{RC}$	2.51	72.43	44.28	60.55	61.50	51.85	73.88	66.69	61.60
	HBLLM$_{row}$	3.25	69.19	46.93	69.60	63.61	56.53	74.92	68.03	64.12
	AF1	2.50	**78.83**	**55.72**	**77.37**	**66.80**	**62.58**	**79.27**	**71.74**	**70.33**
	AF1	1.00	67.85	41.98	64.22	57.11	49.41	72.25	64.48	59.61
Qwen3-32B	BF16	16	83.21	61.09	82.60	67.24	58.04	81.99	72.77	72.42
	RTN	1.00	25.21	26.71	26.37	0.00	0.00	50.00	49.57	25.41
	BiLLM	2.88	63.80	41.30	67.62	62.24	53.13	73.83	66.69	61.23
	STBLLM$_{(4:8)}$	3.50	60.65	37.63	62.42	51.21	43.02	72.58	63.54	55.86
	ARB-LLM$_{RC}$	2.51	75.97	54.35	71.91	**71.28**	62.62	78.78	71.98	69.56
	HBLLM$_{row}$	3.25	75.17	50.34	75.63	66.50	56.16	76.99	70.17	67.28
	AF1	2.50	**81.23**	**59.56**	**80.28**	70.54	**62.97**	**80.52**	**72.30**	**72.49**
	AF1	1.00	71.72	46.25	70.37	61.61	55.13	75.35	69.38	64.26
Gemma-3-4B-it	BF16	16	78.20	56.74	74.09	59.17	54.49	77.15	70.01	67.12
	RTN	1.00	25.02	26.15	25.73	0.00	0.00	50.45	49.60	25.28
	BiLLM	2.88	36.57	24.91	35.29	9.33	6.64	60.01	53.59	32.33
	STBLLM$_{(4:8)}$	3.50	33.88	22.18	31.34	5.67	4.68	55.28	52.17	29.31
	ARB-LLM$_{RC}$	2.51	49.20	29.10	40.62	30.35	23.33	65.61	54.62	41.83
	HBLLM$_{row}$	3.25	60.40	36.52	54.13	41.68	33.01	69.42	57.30	50.35
	AF1	2.50	**71.46**	**45.05**	**68.47**	**57.89**	**51.78**	**76.06**	**65.35**	**62.29**
	AF1	1.00	57.74	35.60	49.83	42.49	37.58	67.35	58.43	49.86
Gemma-3-12B-it	BF16	16	76.73	60.32	81.89	67.71	65.61	78.51	74.66	72.20
	RTN	1.00	25.33	26.01	25.88	0.00	0.00	50.23	49.66	25.30
	BiLLM	2.88	43.60	28.67	41.82	22.06	13.97	62.79	55.96	38.41
	STBLLM$_{(4:8)}$	3.50	34.01	24.06	33.57	13.45	8.52	54.68	51.93	31.46
	ARB-LLM$_{RC}$	2.51	46.93	29.27	45.98	50.73	34.60	64.64	57.93	47.15
	HBLLM$_{row}$	3.25	67.80	43.69	67.00	43.92	37.61	74.70	67.09	57.40
	AF1	2.50	**80.35**	**59.30**	**76.99**	**64.39**	**61.28**	**79.27**	**71.51**	**70.44**
	AF1	1.00	66.95	41.86	57.42	54.45	47.81	77.36	67.30	59.02

Table 4: Detailed zero-shot accuracy on ARC-Easy (AE), ARC-Challenge (AC), HellaSwag (HS), LAMBADA-openai (LO), LAMBADA-standard (LS), PIQA (PQ), and WinoGrande (WG). The average score is computed over the seven zero-shot tasks. Bold and underline denote the best and second-best quantized results within each model.

η	LLaMA-3-8B	Qwen3-14B	Gemma-3-4B-it	Avg.
0.01	**22.18**	**13.32**	<u>32.82</u>	**22.77**
0.02	<u>22.35</u>	<u>13.45</u>	**32.74**	<u>22.85</u>
0.03	22.51	13.58	33.02	23.04
0.04	22.43	13.72	33.11	23.09
0.05	22.69	13.66	33.23	23.19
0.06	22.58	13.84	33.34	23.25
0.07	22.83	13.79	33.50	23.37
0.08	22.74	14.01	33.61	23.45
0.09	23.05	13.95	33.70	23.57

Table 5: Ablation on the residual-energy threshold η used in the adaptive null-space cutoff. Results are reported as WikiText2 perplexity on LLaMA-3-8B, Qwen3-14B, and Gemma-3-4B-it. Bold and underline denote the best and second-best results, respectively.

Model	**Avg. Bits(W)**	**General Allocation Baselines (PPL↓)**					**HiSA Variants (PPL↓)**		
		Uniform	**LIM**	**ZD**	**Act.**	**NLL**	**Only Dir.**	**Only Mag.**	**HiSA**
LLaMA-3-8B	1.00	35.84	29.67	32.51	27.94	25.86	24.59	23.33	**22.18**
Qwen3-8B	1.00	28.92	23.38	25.71	21.45	19.37	18.15	17.59	**16.26**

Table 6: Ablation on different allocation-sensitivity metrics and HiSA variants under the same 1.00-bit weight budget. We report WikiText2 perplexity (PPL; lower is better). Only Dir. and Only Mag. use only directional and magnitude distortion, respectively, while HiSA uses the full Soft-OR fusion of both distortion modes. Bold denotes the best result.

Prompt	Method	Response	Assessment
Q1: I have 3 apples, buy 5 more, and give away 2. How many apples are left? **Answer:** 6	BF16	You start with 3 apples, buy 5 more, and give away 2. Final answer: 6 apples.	Correct
	BiLLM	The ** ** ** ** ** ** ** ** ** ** ** ...	Degenerated
	ARB-LLM	$3 + 5 = 8$, and $8 - 2 = 6$. 6 apples are left.	Correct
	HBLLM	$3 + 5 = 8$ apples, then $8 - 2 = 6$ apples.	Correct
	AF1	You start with 3 apples, buy 5 more, making 8 apples. Then you give away 2, so you have 6 apples left.	Correct
Q2: What are the capitals of Canada, Japan, and Germany? **Answer:** Ottawa, Tokyo, Berlin	BF16	Canada: Ottawa; Japan: Tokyo; Germany: Berlin.	Correct
	BiLLM	Capital of Canada: Montreal. ** ** ** ** ** ...	Wrong + repetitive
	ARB-LLM	Canada's capital is Ottawa, Japan's capital is Tokyo, and Germany's capital is Berlin.	Correct
	HBLLM	Canada: Ottawa; Japan: Tokyo; Germany: Berlin.	Correct
	AF1	Canada: Ottawa; Japan: Tokyo; Germany: Berlin.	Correct
Q3: A train travels 120 kilometers in 2 hours. What is its average speed? **Answer:** 60 km/h	BF16	Average speed is $120/2 = 60$ km/h.	Correct
	BiLLM	60 km/h, followed by ** ** ** ** ** ...	Correct but repetitive
	ARB-LLM	The average speed is 60 kilometers per hour.	Correct
	HBLLM	Average speed is computed as distance divided by time, giving 60 km/h.	Correct
	AF1	Average speed $= 120/2 = 60$ kilometers per hour.	Correct
Q4: Which planet is known as the Red Planet? **Answer:** Mars	BF16	The planet known as the Red Planet is Mars.	Correct
	BiLLM	The Red Planet is not commonly used in real-world science ··· repeated	Wrong + repetitive
	ARB-LLM	The planet known as the Red Planet is Mars, also known as Mars or the Red Planet.	Correct but redundant
	HBLLM	The planet known as the Red Planet is Mars.	Correct
	AF1	The planet known as the "Red Planet" is Mars.	Correct
Q5: What is the sum of the interior angles of a triangle? **Answer:** 180 degrees	BF16	The sum of the interior angles of a triangle is 180 degrees.	Correct
	BiLLM	******** ** ** ** ** ** ** ** ...	Symbol collapse
	ARB-LLM	The sum of the interior angles of a triangle is 180 degrees.	Correct
	HBLLM	The sum of the interior angles of a triangle is 180 degrees.	Correct
	AF1	The sum of the interior angles of a triangle is 180 degrees.	Correct
Q6: What is the chemical formula of water? **Answer:** H_2O	BF16	The chemical formula of water is H_2O.	Correct
	BiLLM	The formula is H_2O, but it repeatedly states that water is composed of two water molecules ···	Wrong explanation + repetition
	ARB-LLM	The formula is H_2O. Water is composed of two hydrogen atoms and one oxygen atom ··· repeated.	Correct but repetitive
	HBLLM	The chemical formula of water is H_2O.	Correct
	AF1	The formula is H_2O, but the explanation is repeated afterwards.	Correct but repetitive

Table 7: Qualitative examples on Qwen3-14B-Instruct under full precision and different binary PTQ methods. All prompts have objective answers. Dark green marks correct key information, dark red marks wrong answers or degenerated repetitions, and dark orange indicates partially correct but unstable generations. Repeated continuations are abbreviated for readability.