

1 A Introducing Haar Wavelet Transform into LLM Quantization

2 Haar wavelet transform converts an original vector into a new matrix containing low- and high-
3 frequency information by computing averages and differences. This transformation can be extended
4 to larger vectors and multidimensional data and is commonly used in image processing and signal
5 analysis.

6 Applying the Haar wavelet transform to LLM quantization offers three advantages:

- 7 • The inverse quantization set becomes richer in representation, which can be demonstrated via the
8 CIQ metric.
- 9 • The data distribution becomes more concentrated: with approximately 65% probability, the
10 variance of the high- and low-frequency coefficient sets in each row is smaller than before the
11 transformation.
- 12 • The additional computational cost for inference is $\mathcal{O}(d)$, where d is the input length, as the
13 Haar transform can be implemented using local convolutional layers, resulting in lower cost than
14 methods such as FrameQuant.

15 A.1 Definition of Haar Transform

16 Let the one-dimensional input signal be x . The output after wavelet transformation is:

$$\hat{x} := [a_1, b_1, \dots, a_i, b_i], \quad (1)$$

17 where a_i denotes low-frequency coefficients and b_i denotes high-frequency coefficients.

18 This process can also be expressed in matrix form:

$$\hat{x}^T = \mathcal{H}(x) := \mathbf{U}_{\text{diag}} x^T, \quad (2)$$

19 where

$$\mathbf{U}_{\text{diag}} := \begin{pmatrix} \mathbf{U} & & & \\ & \mathbf{U} & & \\ & & \ddots & \\ & & & \mathbf{U} \end{pmatrix}. \quad (3)$$

20 Because the Haar matrix is orthogonal, the inverse Haar transform is:

$$x^T = \mathbf{U}_{\text{diag}}^T \hat{x}^T. \quad (4)$$

21 A.1.1 Row-wise and Column-wise Haar Transforms on Matrices

22 Define row-wise and column-wise Haar transforms of a matrix \mathbf{W} as:

$$\mathcal{H}_{\text{row}}(\mathbf{W}) := \mathbf{W} \mathbf{U}_{\text{diag}}^T, \quad \mathcal{H}_{\text{col}}(\mathbf{W}) := \mathbf{U}_{\text{diag}} \mathbf{W}. \quad (5)$$

23 Below is a simple example showing how to apply Haar transforms to a matrix.

24 Given a 4×4 matrix defined as:

$$\mathbf{A} = \begin{pmatrix} 16 & 18 & 22 & 20 \\ 12 & 14 & 10 & 8 \\ 24 & 26 & 30 & 28 \\ 20 & 22 & 18 & 16 \end{pmatrix}, \quad (6)$$

25 the result after applying row-wise Haar transform is denoted as

$$\hat{\mathbf{A}}_{\text{row}} = \sqrt{2} \times \begin{pmatrix} 17 & -1 & 21 & 1 \\ 13 & -1 & 9 & 1 \\ 25 & -1 & 29 & 1 \\ 21 & -1 & 17 & 1 \end{pmatrix}, \quad (7)$$

26 and after applying column-wise Haar transform is denoted as

$$\hat{\mathbf{A}}_{\text{col}} = \sqrt{2} \times \begin{pmatrix} 15 & 0 & 23 & 0 \\ -2 & 0 & -2 & 0 \\ 15 & 0 & 23 & 0 \\ -2 & 0 & -2 & 0 \end{pmatrix}. \quad (8)$$

27 where each element of matrices is multiplied by the coefficient $\sqrt{2}$.

28 **B Relationship Between Richness of Inverse Quantization Set and Model** 29 **Fidelity**

30 **B.1 Definition and Application of CIQ Metric**

31 The cardinality of an inverse quantization set (CIQ) measures the number of distinct values that can
32 be recovered from quantized weights in each row. For linear quantization without partitioning, CIQ
33 equals the bit-width of quantized weights, since linear quantization evenly distributes several points
34 across the original data range. When partitioning strategies are applied, it characterizes expressiveness
35 of the quantization algorithm.

36 In HBLLM, which follows the GPTQ quantization scheme, quantization is done per row of each
37 matrix block. We define CIQ in terms of a single row of a matrix block in the following discussion
38 without loss of generality. Furthermore,

$$CIQ = \min\{\text{row length, maximum recovery ability under given quantization parameters}\}.$$

39 holds.

40 To study the composition of IQ, we introduce a mapping from the set of quantized weights to the set
41 of dequantized weights. Several mappings from quantized weights to inverse quantized values exist:

- 42 • Identity mapping
- 43 • Group merging mapping
- 44 • Residual merging mapping
- 45 • Inverse transformation mapping

46 For example, an inverse quantization set brought by residual merging mapping is defined as:

$$\mathbf{IQ}_{\text{residual}} = \text{inv}_{\text{residual}}(\mathbf{X}_1, \mathbf{X}_2) := \{z : z = (x + y), \forall x \in \mathbf{X}_1, \forall y \in \mathbf{X}_2\}. \quad (9)$$

47 Furthermore,

$$CIQ_{\text{residual}} \leq |\mathbf{X}_1| \cdot |\mathbf{X}_2|, \quad (10)$$

48 where $|\cdot|$ is the cardinality of a set.

49 An inverse quantization set brought by Haar inverse transformation is defined as:

$$\mathbf{IQ}_{\text{Haar}} = \text{inv}_{\text{Haar}}(\hat{\mathbf{X}}_{\text{low}}, \hat{\mathbf{X}}_{\text{high}}) := \left\{ z : z = \frac{1}{\sqrt{2}}(x + y) \text{ or } z = \frac{1}{\sqrt{2}}(x - y), \forall x \in \hat{\mathbf{X}}_{\text{low}}, \forall y \in \hat{\mathbf{X}}_{\text{high}} \right\}. \quad (11)$$

50 And then,

$$CIQ_{\text{Haar}} \leq 2|\hat{\mathbf{X}}_{\text{low}}| \cdot |\hat{\mathbf{X}}_{\text{high}}|. \quad (12)$$

51 Let an inverse quantization set produced by BiLLM algorithm be denoted as $\mathbf{IQ}_{\text{BiLLM}}$. According to
52 BiLLM algorithm, $\mathbf{IQ}_{\text{BiLLM}}$ can be expressed in the following form:

$$\begin{aligned} \mathbf{IQ}_{\text{BiLLM}} &= \mathbf{IQ}_{\text{residual}}^{\text{sal}} \cup \mathbf{IQ}_{\text{residual}}^{\text{non-sal}} \\ &= \text{inv}_{\text{residual}}(\mathbf{X}_1^{\text{sal}}, \mathbf{X}_2^{\text{sal}}) \cup \mathbf{X}_1^{\text{non-sal}} \cup \mathbf{X}_2^{\text{non-sal}}. \end{aligned} \quad (13)$$

53 where $\mathbf{IQ}_{\text{residual}}^{\text{sal}}$ and $\mathbf{IQ}_{\text{residual}}^{\text{non-sal}}$ represent inverse quantization sets of the salient and non-salient parts,
54 respectively.

55 **Lemma 1.** *BiLLM has at most 8 different dequantized values per row.*

56 *Proof.* According to (13) and the definition of CIQ, we can infer that

$$CIQ_{\text{BiLLM}} \leq CIQ_{\text{residual}}^{\text{sal}} + |\mathbf{X}_1^{\text{non-sal}}| + |\mathbf{X}_2^{\text{non-sal}}|. \quad (14)$$

57 Since the salient part adopts a residual approximation strategy, it follows that

$$CIQ_{\text{residual}}^{\text{sal}} \leq |\mathbf{X}_1|^{\text{sal}} \cdot |\mathbf{X}_2|^{\text{sal}}. \quad (15)$$

58 Given that $|\mathbf{X}_1^{\text{sal}}| = |\mathbf{X}_2^{\text{sal}}| = |\mathbf{X}_1^{\text{non-sal}}| = |\mathbf{X}_2^{\text{non-sal}}| = 2$, substituting (15) into (14) yields:

$$CIQ_{\text{BiLLM}} \leq |\mathbf{X}_1^{\text{sal}}| |\mathbf{X}_2^{\text{sal}}| + |\mathbf{X}_1^{\text{non-sal}}| + |\mathbf{X}_2^{\text{non-sal}}| = 2 \times 2 + 2 + 2 = 8. \quad (16)$$

59 □

60 Next, we analyze theoretical upper bounds of CIQ for HBLLM algorithms. We first consider the case
61 of HBLLM-col. Let an inverse quantization set produced by HBLLM-col be denoted as $\mathbf{IQ}_{\text{HBLLM-col}}$.
62 According to HBLLM-col algorithm, $\mathbf{IQ}_{\text{HBLLM-col}}$ can be expressed in the following form:

$$\begin{aligned} \mathbf{IQ}_{\text{HBLLM-col}} &= \mathbf{IQ}_{\text{Haar}}^{\text{sal}} \cup \mathbf{IQ}_{\text{Haar}}^{\text{non-sal}} \\ &= \text{inv}_{\text{Haar}} \left(\hat{\mathbf{X}}_{\text{low}}^{\text{sal}}, \hat{\mathbf{X}}_{\text{high}}^{\text{sal}} \right) \cup \text{inv}_{\text{Haar}} \left(\hat{\mathbf{X}}_{\text{low}}^{\text{non-sal}}, \hat{\mathbf{X}}_{\text{high}}^{\text{non-sal}} \right), \end{aligned} \quad (17)$$

63 where $\mathbf{IQ}_{\text{Haar}}^{\text{sal}}$ and $\mathbf{IQ}_{\text{Haar}}^{\text{non-sal}}$ represent the dequantized sets of the salient and non-salient parts, respec-
64 tively. Both parts employ a group quantization strategy under Haar transform, so the upper bound for
65 each part is the product of the cardinality of the two Haar sub-band quantization sets. Additionally,
66 within HBLLM-col algorithm, each sub-band has two groups, resulting in a total of four quantized
67 values. Based on the above analysis, we arrive at the second conclusion.

68 **Lemma 2.** *HBLLM-col has at most 64 different dequantized values per row.*

Proof.

$$CIQ_{\text{HBLLM-col}} \leq 2 \left| \hat{\mathbf{X}}_{\text{low}}^{\text{sal}} \right| \left| \hat{\mathbf{X}}_{\text{high}}^{\text{sal}} \right| + 2 \left| \hat{\mathbf{X}}_{\text{low}}^{\text{non-sal}} \right| \left| \hat{\mathbf{X}}_{\text{high}}^{\text{non-sal}} \right| = 64. \quad (18)$$

69 □

70 The CIQ upper bound of HBLLM-row algorithm is significantly larger than that of HBLLM-col
71 algorithm. Let an inverse quantization set produced by HBLLM-row algorithm be denoted as
72 $\mathbf{IQ}_{\text{HBLLM-row}}$. By HBLLM-row algorithm, $\mathbf{IQ}_{\text{HBLLM-row}}$ can be shown in the following form:

$$\mathbf{IQ}_{\text{HBLLM-row}} = \mathbf{IQ}_{\text{HBLLM-row}}^{\text{sal}} \cup \mathbf{IQ}_{\text{Haar}}^{\text{non-sal}}, \quad (19)$$

73 where $\mathbf{IQ}_{\text{HBLLM-row}}^{\text{sal}}$ is defined as

$$\mathbf{IQ}_{\text{HBLLM-row}}^{\text{sal}} = \text{inv}_{\text{residual}} \left(\mathbf{IQ}_{\text{Haar}}^{\text{sal}}, \mathbf{IQ}_{\text{Haar}}^{\text{non-sal}} \right). \quad (20)$$

74 Unlike HBLLM-col, the non-salient part of HBLLM-row encompasses the entire matrix area, resulting
75 in overlap with the salient part. Therefore, HBLLM-row incorporates a residual approximation on the
76 salient part, further increasing the CIQ upper bound. Additionally, within HBLLM-row algorithm,
77 each sub-band has two groups, resulting in a total of four quantized values. Based on the above
78 analysis, we arrive at the third conclusion.

79 **Lemma 3.** *HBiLLM-row can have over 1024 different dequantized values per row.*

Proof.

$$\begin{aligned} CIQ_{\text{HBLLM-row}}^{\text{sal}} &\leq \left| \mathbf{IQ}_{\text{Haar}}^{\text{sal}} \right| \left| \mathbf{IQ}_{\text{Haar}}^{\text{non-sal}} \right| \\ &= \left| \text{inv}_{\text{Haar}} \left(\hat{\mathbf{X}}_{\text{low}}^{\text{sal}}, \hat{\mathbf{X}}_{\text{high}}^{\text{sal}} \right) \right| \times \left| \text{inv}_{\text{Haar}} \left(\hat{\mathbf{X}}_{\text{low}}^{\text{non-sal}}, \hat{\mathbf{X}}_{\text{high}}^{\text{non-sal}} \right) \right| \\ &\leq 2 \left| \hat{\mathbf{X}}_{\text{low}}^{\text{sal}} \right| \left| \hat{\mathbf{X}}_{\text{high}}^{\text{sal}} \right| \times 2 \left| \hat{\mathbf{X}}_{\text{low}}^{\text{non-sal}} \right| \left| \hat{\mathbf{X}}_{\text{high}}^{\text{non-sal}} \right| \\ &= 1024. \end{aligned} \quad (21)$$

$$CIQ_{\text{HBLLM-row}} \leq 1024 + 32 = 1056. \quad (22)$$

80 □

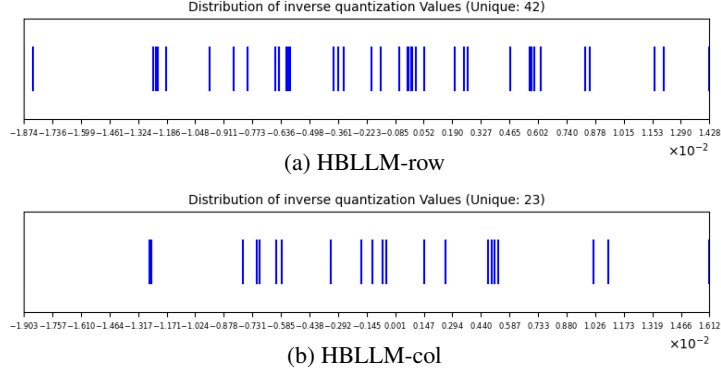


Figure B.1: CIQ and Distribution of inverse quantization Values

Lemma 3 is a result that holds under the assumption that there are sufficiently many columns in the salient part. In practical algorithms, due to the dual constraints of the quantized matrix size and the total bitrate, the upper bound of $CIQ_{HBLLM-row}$ is much less than 1024. The specific upper bound can be described in Lemma 4.

Lemma 4. *Let the size of a quantized matrix block be $d \times d$, where $d \leq 256$, and let the proportion of the number of columns in the salient part to the total number of columns be p (where $0 < p < 1$). Then, we have*

$$CIQ_{HBLLM-row} \leq 32 + p \cdot d, \quad (23)$$

and

$$CIQ_{HBLLM-col} \leq 32 + \min\{p \cdot d, 32\}. \quad (24)$$

Proof. It is easy to get by Lemma 2 and Lemma 3. \square

Theorem 1. *Under the same proportion p of the salient part, $CIQ_{HBLLM-row} \geq CIQ_{HBLLM-col}$ holds.*

Proof. This follows from Lemma 4. \square

B.2 Measured Values of the CIQ Metric for the HBLLM Algorithm

To validate the theoretical analysis of CIQ, we present two representative examples under the HBLLM-row and HBLLM-col schemes, respectively. These examples illustrate how the CIQ values measured in practice align with the theoretical bounds derived earlier.

Example 1: HBLLM-row. As shown in B.1a, consider a row of length 128 where 14 elements are marked as salient. After quantization and reconstruction, the total number of distinct values observed is 42. Among these, 30 values originate from the non-salient part (i.e., $|\mathbf{IQ}_{Haar}^{non-sal}| = 30$) and 7 values from the salient part (i.e., $|\mathbf{IQ}_{Haar}^{sal}| = 7$).

Although the salient part involves residual merging, the final reconstructed set still satisfies:

$$CIQ_{HBLLM-row} = 42 \leq 30 + 14 = 44, \quad (25)$$

which confirms that the practical CIQ value stays within the theoretical upper bound in Lemma 4.

Example 2: HBLLM-col. As shown in B.1b, In a second example under the HBLLM-col scheme, a row consists of 119 non-salient and 9 salient elements. The measured CIQ is 23, with 16 values from the non-salient part and 7 values from the salient part. This result again satisfies:

$$CIQ_{HBLLM-col} = 23 \leq 16 + 9 = 25. \quad (26)$$

These observations demonstrate that in practice, the effective size of the inverse quantization set is significantly below the worst-case bounds, especially when some quantized values are shared or overlap. They also confirm the effectiveness of the HBLLM decomposition strategies in maintaining a compact and expressive representation of quantized weights.

B.3 Limitations of CIQ Metric Analysis and the Necessity of Introducing a Structure-Aware Grouping Strategy

Although HBLLM significantly outperforms BiLLM in the CIQ metric, it lacks a clear advantage in quantization performance without the structure-aware grouping strategy introduced in this paper.

To assess the performance differences before and after implementing this strategy, we conduct experiments evaluating perplexity and QA accuracy. Experiment data can be found in Table B.1:

- **BiLLM+ ℓ_2^\dagger** : Employing ℓ_2 -based saliency-driven column selection together with multi-parameter intra-row grouping.
- **Haar+BiLLM**: This refers to a method obtained by removing the ℓ_2 -norm-based saliency-driven column selection and multi-parameter intra-row grouping strategies from HBLLM. This method integrates the one-dimensional discrete Haar wavelet transform applied row-wise or column-wise, thereby deriving two approaches: Row-Haar+BiLLM and Col-Haar+BiLLM, respectively.
- **DCT+BiLLM**: This approach applies the BiLLM algorithm to coefficient matrices obtained from the one-dimensional Discrete Cosine Transform (DCT) applied row-wise or column-wise on weight matrices, resulting in Row-DCT+BiLLM and Col-DCT+BiLLM. Unlike the Haar+BiLLM method, DCT+BiLLM uses a global transformation strategy by first mapping the entire matrix to the Fourier domain before quantization. In contrast, Haar+BiLLM applies the Haar transform to matrix blocks, with quantization following the BiLLM process, making it a local orthogonal transformation.
- **HBiLLM+**: These method, refer to those derived from Haar+BiLLM, utilizing the strategies proposed in our paper. They include HBLLM-row+ ℓ_2 , HBLLM-col+ ℓ_2 , HBLLM-col+ ℓ_2^\dagger and HBLLM-col+ ℓ_2^\dagger .

Table B.1: Perplexity (\downarrow , C4, Wiki2, PTB) and AvgQA accuracy (\uparrow , AvgQA over 9 zero-shot tasks) of BiLLM variants with Haar.

Method	OPT-1.3B				LLaMA2-7B			
	C4 \downarrow	Wiki2 \downarrow	PTB \downarrow	AvgQA \uparrow	C4 \downarrow	Wiki2 \downarrow	PTB \downarrow	AvgQA \uparrow
BiLLM	56.24	68.43	119.2	38.39	33.97	31.38	373.0	42.11
BiLLM+ ℓ_2	55.95	72.42	105.9	37.95	33.46	31.34	695.8	41.11
BiLLM+ ℓ_2^\dagger	56.88	70.48	92.16	39.28	28.17	25.08	226.3	41.77
Row-Haar+BiLLM	47.45	52.81	62.81	39.57	25.77	25.12	138.0	44.67
Col-Haar+BiLLM	95.56	128.8	171.3	36.92	41.03	37.25	5193	39.60
Row-DCT+BiLLM	8010	11517	6729	31.36	45358	49395	26888	34.48
Col-DCT+BiLLM	107.1	150.5	250.1	34.19	26.54	24.64	1202	44.82
HBLLM-row+ ℓ_2	26.47	33.68	41.17	41.17	16.26	19.86	87.90	47.90
HBLLM-col+ ℓ_2	26.37	29.99	36.24	42.13	15.04	13.99	154.6	49.38
HBLLM-row+ ℓ_2^\dagger	19.55	26.95	25.70	46.87	13.00	13.20	85.50	50.86
HBLLM-col+ ℓ_2^\dagger	21.98	23.69	27.39	45.21	13.18	12.02	146.1	51.34

Note: ℓ_2 denotes activation of ℓ_2 -norm-based saliency-driven column selection; \dagger denotes activation of frequency-aware multi-parameter intra-row grouping.

The main experimental results are summarized as follows:

- Directly applying Haar transform into BiLLM pipeline does not significantly improve 1-bit quantization performance.
 - Row-Haar+BiLLM shows slight improvement.
 - Col-Haar+BiLLM decreases performance.
- Implementing the 'saliency-driven column selection via ℓ_2 norm' strategy leads to:
 - Significant improvements for HBLLM-row+ ℓ_2 and HBLLM-col+ ℓ_2 compared to their predecessors.
 - Perplexity tests show:
 - * 32-64% reductions on C4 and Wiki2 test sets.
 - * HBLLM-col+ ℓ_2 shows notable improvements, but still lags behind BiLLM on the PTB test set.

- QA testing accuracy improves by 3-10%.
 - Further introducing ‘frequency-aware multi-parameter intra-row grouping’ results in:
 - HBLLM-row+ ℓ_2 +Row-wise-grouping is the best.
 - 26-45% reductions in perplexity on C4 and Wiki2.
 - Significant improvements on the PTB test set, surpassing BiLLM.
 - Cumulative accuracy in QA testing increases by 2-8%.
- This experimental result demonstrates that introducing a structure-aware grouping strategy is essential for effectively combining the Haar transform with the BiLLM algorithm.

B.4 Effectiveness of Haar Transform and the Importance of Local Orthogonality

As shown in Table B.1, although HBLLM integrates multiple strategies, it is important to disentangle the specific contribution of the Haar-based frequency decomposition from other components such as saliency selection and structure-aware grouping. To this end, we conduct dedicated ablation studies to quantify the standalone effectiveness of the Haar transform and contrast it with global orthogonal alternatives such as Discrete Cosine Transform (DCT).

We summarize our key observations below:

- **Effectiveness of Haar Transform:** While incorporating either the ℓ_2 -based saliency selection or the structure-aware grouping alone yields only modest improvements to BiLLM, introducing the Haar transform leads to consistently more substantial gains in both perplexity and QA accuracy. Notably, even under partial activation (e.g., HBLLM-col+ ℓ_2), the models outperform their BiLLM counterparts.
 - Both HBLLM-row+ ℓ_2^\dagger and HBLLM-col+ ℓ_2^\dagger significantly outperform BiLLM and BiLLM+ ℓ_2^\dagger , underscoring the crucial role of Haar in preserving the frequency-domain structure of weights.
 - Row-Haar+BiLLM, even without ℓ_2 -norm-based saliency-driven column selection or grouping, shows consistent performance gains, confirming that Haar decomposition independently enhances quantization representation capacity.
- **Local vs. Global Orthogonal Transforms:** We further analyze the impact of replacing Haar transform with global DCT.
 - Applying the global row-wise DCT results in severe degradation across all benchmarks, with Row-DCT+BiLLM performing significantly worse than BiLLM.
 - Applying the global column-wise DCT offers moderate improvement on LLaMA2-7B; however, Col-DCT+BiLLM still lags behind BiLLM in all OPT-1.3B tests.
 - These results highlight that global transforms struggle to capture local variations in weight distributions, which are effectively preserved by block-wise Haar decomposition.

As shown in Table B.2, global transforms such as DCT also incur substantial computational overhead compared to local Haar transforms:

Table B.2: Time comparison between BiLLM, DCT+BiLLM, and HBLLM on LLaMA-1 with different model sizes. The DCT implementation used in this test is from pytorch.

Method	7B	13B	30B
BiLLM	36 min	71 min	142 min
DCT+BiLLM	211 min	414 min	1012 min
HBLLM	44 min	98 min	173 min

In conclusion: Haar transform independently and robustly contributes to quantization fidelity, even in the absence of auxiliary strategies such as ℓ_2 -norm-based saliency-driven column selection or grouping; local orthogonal transforms like Haar are consistently more effective than global ones like DCT in preserving localized frequency-domain structures—an essential property for stable and expressive 1-bit quantization.

C Analysis of the Correlation Between Data Concentration and Model Fidelity

In this section, we explore the positive correlation between improved data concentration and enhanced model fidelity after quantization, following the application of the Haar transform and structure-aware grouping strategy proposed by HBLLM. This correlation provides a theoretical foundation for the effectiveness of the HBLLM quantization method. We first observe that the concentration of coefficient distribution improves with a 65% probability after applying the Haar row transform. Based on this observation, we mathematically model the probability of variance improvement in data concentration. We then apply this variance improvement probability to the HBLLM-row and HBLLM-col methods to validate the correlation between enhanced data concentration and improved model fidelity after quantization.

C.1 Improvement of Data Concentration by Haar Transform

We discuss the improvement in data concentration resulting from the Haar transform, which is typically described by variance—lower variance indicates higher data concentration. We examine the impact of the Haar row transform on the distribution characteristics of the weight matrix. Specifically, we select a matrix block from the OPT-1.3B model and compare the variance of each original weight vector with the variances of the low-frequency and high-frequency subbands obtained after Haar decomposition. The variances of each row from different methods are arranged in ascending order, as shown in Figure C.1. Notably, approximately 65% of the rows exhibit a variance in at least one subband that is lower than the original value, indicating that the Haar row transform generally enhances data concentration.

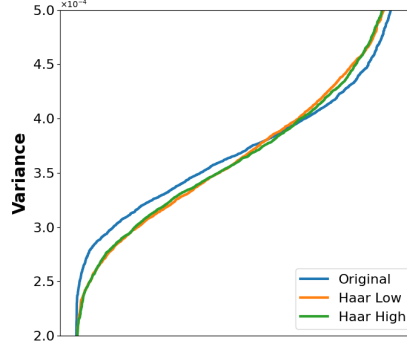


Figure C.1: Row-wise Variance Comparison Before and After Haar Transform

Therefore, we anticipate that the application of HBLLM’s structure-aware grouping strategy may further improve data concentration.

C.2 Mathematical Modeling of Variance Improvement Probability

To quantify the improvement in data concentration achieved by the Haar transform combined with the structure-aware grouping strategy, we introduce a random variable p to describe the probability of improvement in data concentration for each row after applying the strategy, as well as the expected probability of improvement $E(p)$ for each matrix block. Let the matrix block size be $d \times d$, and the j -th row, after the Haar row transform, be divided into M subgroups, each containing n_i^j coefficients, with an intra-group variance of V_i^j . The variance of the entire row before transformation is denoted as \tilde{V}_0 . The sample value p_j for the j -th row is calculated as follows:

$$p_j := \frac{\sum_{i=1}^M n_i^j \times \text{sign}(\max\{0, (\tilde{V}_0 - V_i^j)\})}{d}, \quad (27)$$

where, the sign function $\text{sign}(\cdot)$ takes the value of 1 only when the subgroup variance is less than the original row variance; otherwise, it is 0. The value p_j represents the proportion of coefficients in the j -th row that belong to subgroups with improved concentration. Based on this, we define the expected value $E(p)$ for the entire matrix block to characterize the average level of overall data concentration

improvement:

$$E(p) = \frac{\sum_{j=1}^d p_j}{d}. \quad (28)$$

C.3 Analysis of the Relationship Between Data Concentration and Quantization Fidelity

To further investigate the impact of improved data concentration on model fidelity, we collected data on the probability of variance improvement, the relative ℓ_2 error of matrix blocks before and after quantization, and the corresponding model fidelity, as shown in Figure C.2. The relative ℓ_2 error serves as the optimization criterion for HBLLM quantization, while model fidelity is measured by perplexity—lower perplexity indicates higher fidelity after quantization.

We analyzed the distribution changes of relative ℓ_2 errors for all matrix blocks across different models before and after quantization, and we plotted the perplexity performance under various grouping strategies. The experimental results demonstrate that HBLLM not only significantly enhances data concentration but also effectively mitigates the growth of quantization error, thereby better preserving the model’s original performance.

Figure C.2a displays the proportion $E(p)$ of Haar coefficients across all matrix blocks that meet the variance improvement criterion after combining the row wavelet transform and grouping strategy. The points on the graph represent $E(p)$ for a matrix block in a quantized model, with each graph showing the values of $E(p)$ arranged in ascending order. As illustrated, after applying the strategy, over 65% of Haar coefficients in all matrix blocks achieved an improvement in data concentration, with median improvements of 67% and 68%, respectively.

The results from Figure C.2 lead to the following conclusion: after applying Haar transform combined with the structure-aware grouping strategy proposed by HBLLM, there is a positive correlation between the improvement in data concentration and the enhancement of model fidelity after quantization.

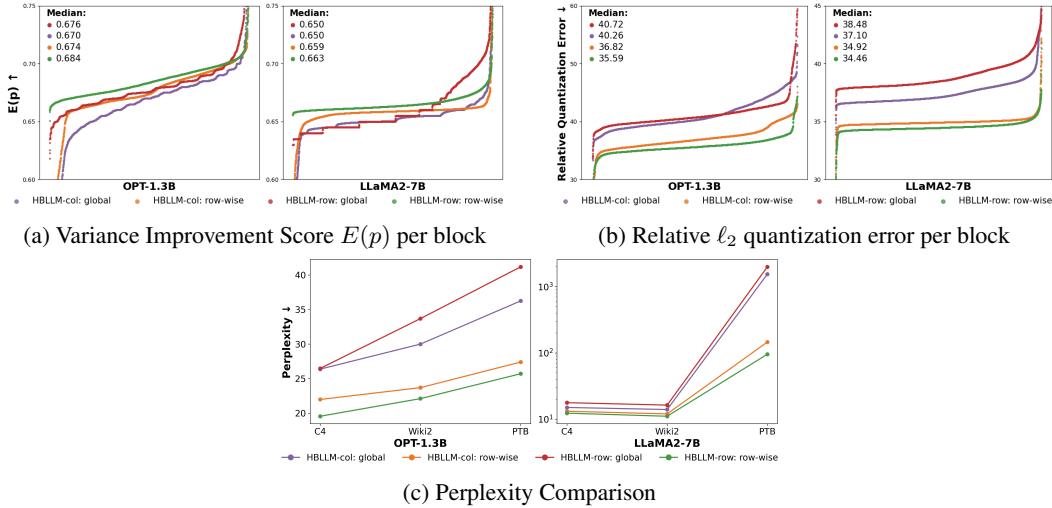
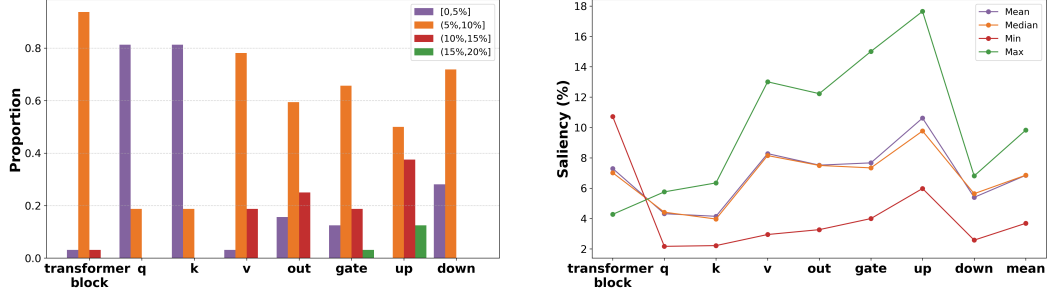


Figure C.2: Comparative Evaluation of Grouping Strategies on Data Concentration, Quantization Error, and Model Perplexity.

C.4 Empirical Analysis of Saliency Ratio Distribution Across Layers and Blocks

To better understand the structure-aware quantization behavior of our method, we analyze how saliency ratio is distributed across different types of layers and transformer blocks. A saliency ratio is defined as the proportion of weights selected by our Hessian-based criterion during quantization.

We visualize the results using two plots: a histogram showing the saliency ratio distribution across different layer types, and a line plot depicting the evolution of block-wise saliency ratio across the network. As shown in Figure C.3a and Figure C.3b, query and key projections exhibit low saliency (mostly below 5%), while value, gate, and up-projection layers tend to have significantly



(a) Saliency Ratio Distribution Across Layers and Blocks

(b) Saliency Statistics Across Layers and Blocks

Figure C.3: The Distribution of Saliency between Different Types of Weight Layers and across Different Transformer Blocks.

higher saliency. Additionally, we observe a gradual increase in saliency ratio from shallow to mid transformer blocks, followed by stabilization.

These results confirm that saliency ratio is not uniformly distributed, but highly dependent on layer type and block depth. This validates our strategy of adapting quantization granularity according to the structure of the model.

D Storage Cost Analysis and Inference Execution Process

To comprehensively assess compression effectiveness, we divide the stored data into three types: weight overhead, coefficient overhead, and flag overhead.

Weight Overhead (W-bits). This refers to the number of bits used to store binarized weights. In standard 1-bit schemes, each weight requires only 1 bit. However, in schemes that retain salient weights, these may be stored with higher precision (e.g., 2-bit or 8-bit), increasing the overall weight overhead.

To increase fidelity, some weights would use more than 1 bit. For example, the salient part employs two 1-bit values with residual approximation. As a result, the average weight overhead per weight (denoted as W-bits) of Billm results would become fractional, such as 1.08 bits.

Coefficient Overhead (C-bits). This refers to the additional bit-width required to store scaling factors and means. For example, OneBit introduces two scaling vectors per row or column, while ARB-LLM_{RC} further computes scaling factors for both rows and columns. Although these parameters are smaller in size than the weight matrix, they must be tightly controlled in precision-critical applications.

Flag Overhead (F-bits). These bits are used to store indicators for salient/non-salient weights (such as the "salient column" tag in PB-LLM or bitmap/group masks in BiLLM), or group affiliation information (e.g., group IDs).

To objectively evaluate the compression efficiency of various methods, we use the "average bit-width per weight" (**Average-Bit**) as a unified metric. This metric characterizes not only storage cost but also the bandwidth required from memory to GPU registers. The average bit-width is computed as:

$$\text{AvgBit} = \frac{\text{Total storage bits}}{\text{Total number of parameters}} \times \text{Structure expansion factor.} \quad (29)$$

here, the total number of parameters refers to the product of the matrix dimensions. The structure expansion factor accounts for mismatches in the number of stored units and original parameters (e.g., 1 for non-restructuring methods, > 1 for methods like FrameQuant).

D.1 Data Distributions of Various Binarization Methods

Figure D.1 and Table D.1 illustrate how different LLM binarization methods distribute their storage data. PB-LLM uses a hybrid-precision quantization strategy that encodes 10% of weights with 8-bit asymmetric linear quantization and binarizes the remaining 90%. A 1-bit flag differentiates salient

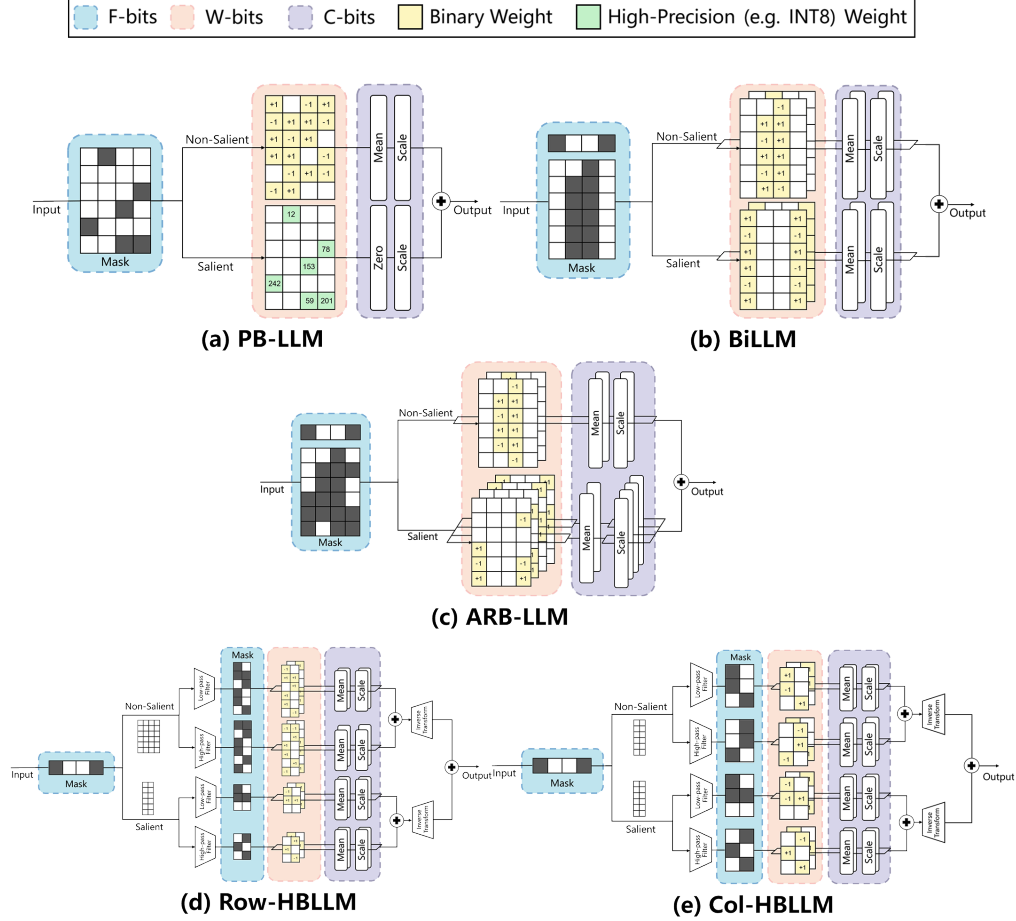


Figure D.1: Overview of storage and inference procedure across different LLM binarization methods.

Table D.1: Storage data composition of different LLM binarization methods.

Method	W-Bits	F-Bits	C-Bits	Average-Bits
PB-LLM	1.70	1.000	0.500	3.200
BiLLM	1.08	1.008	0.875	2.963
ARB-LLM	1.08	1.008	1.25	3.338
HBLLM-row	1.08	1.088	1.25	3.418
HBLLM-col	1.00	1.008	0.875	2.883

from non-salient weights. Non-salient parts are recovered using shared scaling and mean vectors, while salient weights use separate parameters and zero-points. All coefficients are stored in FP16 and shared per output channel, resulting in 0.5 bits of coefficient overhead per weight. PB-LLM’s AvgBit is 3.2 bits.

BiLLM extends basic binarization with residual approximation for salient weights. Columns are divided into salient and non-salient parts. 90% of weights use standard 1-bit encoding, while the salient portion is enhanced with residual binarization. This yields a weight overhead of 1.08 bits per weight. Additional structure information (bitmap, grouping) accounts for 1.008 bits per weight. With FP16-stored coefficients and two sets of residual parameters for salient parts, total coefficient overhead is 0.875 bit per weight. Hence, BiLLM’s AvgBit is 2.963 bits.

ARB-LLM introduces group modeling for salient columns based on BiLLM to better capture complex distributions. weight overhead remains 1.08 bits per weight. Structure metadata (CGB) takes 1.008 bits per weight. All coefficients are stored in FP16. Salient columns are grouped and assigned two sets of second-order coefficients, bringing coefficient overhead to 1.25 bits per weight. ARB-LLM’s AvgBit reaches 3.338 bits.

HBLLM extends BiLLM by introducing structure-aware grouping strategies to improve quantization fidelity in the Haar domain. The weight matrix is split into salient and non-salient parts. Salient columns undergo column-wise Haar transforms, while non-salient parts are transformed along either row or column directions, followed by grouped binarization. All weights are quantized using standard 1-bit encoding. HBLLM-col has a weight overhead of 1.00 bit per weight; HBLLM-row employs a neighborhood averaging strategy (FillAvg) to reconstruct missing values, increasing the weight overhead to 1.08 bits/weight. Saliency bitmaps and frequency-aware grouping metadata add 1.008 bits per weight. All reconstruction coefficients are stored in FP16. HBLLM-row forms four subgroups per row with independent scale and mean values, resulting in a coefficient overhead of 1.25 bits per weight. HBLLM-col shares four subgroups across two rows, averaging two groups per row, and applies intra-band mean sharing to reduce coefficient overhead to 0.875 bit per weight. The final average bit-widths of HBLLM-row and HBLLM-col are 3.418 and 2.883 bits, respectively.

D.2 Details of Average Bit-Width Calculation

The average bit-width of a quantized matrix $\widehat{\mathbf{W}} \in \mathbb{R}^{n \times m}$ is defined as the total memory cost (in bits) divided by the number of elements in the original matrix:

$$\text{AvgBit} = \frac{\mathcal{M}}{n \times m}. \quad (30)$$

For $\mathbf{W} \in \mathbb{R}^{n \times m}$, block size k , the memory of $\widehat{\mathbf{W}}$ after standard row-wise binarization is

$$\mathcal{M}^{\text{1st}} = \underbrace{n \times m}_{\mathbf{B}} + \underbrace{\lceil m/k \rceil}_{\text{multiple blocks}} \times \underbrace{2 \times n \times 16}_{\text{row-wise FP16 } \alpha \text{ and } \mu}. \quad (31)$$

Moreover, second-order row-wise binarization can be represented as

$$\mathcal{M}^{\text{2nd}} = \underbrace{2 \times n \times m}_{\mathbf{B}_1 \text{ and } \mathbf{B}_2} + \underbrace{\lceil m/k \rceil}_{\text{multiple blocks}} \times \underbrace{3 \times n \times 16}_{\text{row-wise FP16 } \alpha_1, \alpha_2, \text{ and } \mu}, \quad (32)$$

since row-wise μ_1 and μ_2 can be combined together as $\mu = \mu_1 + \mu_2$. Thus, the memory required by BiLLM can be formulated as

$$\mathcal{M}_{\text{BiLLM}} = \underbrace{2 \times n \times c + \lceil m/k \rceil \times 3n \times 16}_{\text{second-order binarization}} \quad (33)$$

$$+ \underbrace{n \times (m - c) + \lceil m/k \rceil \times 2n \times 16 \times 2}_{\substack{\text{first-order binarization} \\ \text{2 groups}}} + \underbrace{n \times m}_{\text{group bitmap}} + \underbrace{\widetilde{m}}_{\text{salient column bitmap}}, \quad (34)$$

where c is the number of salient columns for \mathbf{W} .

Similarly, we can formulate the memory occupation of first-order row-column-wise binarization and ARB-RC as

$$\mathcal{M}_{\text{ARB-RC}} = \underbrace{2 \times n \times c + (\lceil m/k \rceil \times 2n + 2c) \times 16}_{\substack{\text{second-order binarization} \\ \text{2 groups}}} \quad (35)$$

$$+ \underbrace{n \times (m - c) + (\lceil m/k \rceil \times n + (m - c)) \times 16 \times 2}_{\substack{\text{first-order binarization} \\ \text{2 groups}}} + \underbrace{n \times m}_{\text{group bitmap}} + \underbrace{\widetilde{m}}_{\text{salient column bitmap}}. \quad (36)$$

In addition, since CGB is used in the experiments, the total memory of ARC-RC + CGB is

$$\mathcal{M}_{\text{ARB-RC+CGB}} = \underbrace{2 \times n \times c + (\lceil m/k \rceil \times 2n + 2c) \times 16 \times 2}_{\substack{\text{second-order binarization} \\ \text{2 groups}}} \quad (37)$$

$$+ \underbrace{n \times (m - c) + (\lceil m/k \rceil \times n + (m - c)) \times 16 \times 2}_{\text{first-order binarization, 2 groups}} + \underbrace{n \times m}_{\text{group bitmap}} + \underbrace{\tilde{m}}_{\text{salient column bitmap}}. \quad (38)$$

Furthermore, we formulate the memory cost of PBLLM by considering both unsalient weights and salient weights as

$$\mathcal{M}_{\text{PBLLM}} = \underbrace{r_{\text{binary}} \times n \times m + \lceil m/k \rceil \times 2n \times 16}_{\text{unsalient weights}} \quad (39)$$

$$+ \underbrace{(1 - r_{\text{binary}}) \times n \times m \times 8 + \lceil m/k \rceil \times 2n \times 16}_{\text{salient weights}} + \underbrace{n \times m}_{\text{group bitmap}}, \quad (40)$$

where r_{binary} denotes the ratio of the binarized weights.

HBLLM-row adopts four subgroups per row with independent α and μ , intra-band mean sharing, and a neighborhood-based reconstruction strategy (FillAvg), which increases the group bitmap cost.

$$\mathcal{M}_{\text{HBLLM-row}} = \underbrace{n \times m + \lceil m/k \rceil \times 3n \times 16 \times 2}_{\text{unsalient weights, 2 groups}} \quad (41)$$

$$+ \underbrace{n \times c + \lceil m/k \rceil \times 2n \times 16 \times 2}_{\text{salient weights, 2 groups}} + \underbrace{n \times (m + c)}_{\text{group bitmap}} + \underbrace{\tilde{m}}_{\text{salient column bitmap}}. \quad (42)$$

HBLLM-col shares four subgroups across two rows and applies intra-band mean sharing.

$$\mathcal{M}_{\text{HBLLM-col}} = \underbrace{n \times (m - c) + \lceil m/k \rceil \times 1.5n \times 16 \times 2}_{\text{unsalient weights, 2 groups}} \quad (43)$$

$$+ \underbrace{n \times c + \lceil m/k \rceil \times 2n \times 16 \times 2}_{\text{salient weights, 2 groups}} + \underbrace{n \times m}_{\text{group bitmap}} + \underbrace{\tilde{m}}_{\text{salient column bitmap}}. \quad (44)$$

Example: Average Bit-width of HBLLM-row and HBLLM-col

Assume $\mathbf{W} \in \mathbb{R}^{n \times m}$, block size $k = 128$ and the number of salient columns is $c = 0.08m$. The total memory cost for HBLLM-row can be expressed as:

$$\mathcal{M}_{\text{HBLLM-row}} = nm + \left\lceil \frac{m}{k} \right\rceil \cdot 3n \cdot 16 \cdot 2 + nc + \left\lceil \frac{m}{k} \right\rceil \cdot 2n \cdot 16 \cdot 2 + n(m + c) + \tilde{m} \quad (45)$$

$$= 2nm + 2nc + 160n \left\lceil \frac{m}{k} \right\rceil + \tilde{m}. \quad (46)$$

Thus, the average bit-width is:

$$\text{AvgBit} = \frac{\mathcal{M}_{\text{HBLLM-row}}}{nm} = 2.16 + \frac{160n \left\lceil \frac{m}{k} \right\rceil}{nm} + \frac{\tilde{m}}{nm} \approx 3.418 \text{ bits}. \quad (47)$$

Similarly, for HBLLM-col:

$$\mathcal{M}_{\text{HBLLM-col}} = n(m - c) + \left\lceil \frac{m}{k} \right\rceil \cdot 1.5n \cdot 16 \cdot 2 + nc + \left\lceil \frac{m}{k} \right\rceil \cdot 2n \cdot 16 \cdot 2 + nm + \tilde{m} \quad (48)$$

$$= 2nm + 112n \left\lceil \frac{m}{k} \right\rceil + \tilde{m}. \quad (49)$$

Algorithm E.1 HBLLM: Detailed functions process

func SALIENT(\mathbf{W}, \mathbf{H}^c)

```

1:  $\mathbf{S} \leftarrow \mathbf{W}^2 / [\mathbf{H}_{b:b+\beta, b:b+\beta}^c]^2$  // salient matrix
2:  $\text{rows}\{\cdot\} \leftarrow \text{topk}(\|\mathbf{S}\|_2, \text{dim} = 0)$ 
3:  $e \leftarrow \infty$  // searching error
4:  $K \leftarrow 0$  // optimal number of salient columns
5: for  $i = 1, 2, \dots, \text{len}(\text{rows})$  do
6:    $\mathbf{B}_1 \leftarrow \text{BINARY}(\mathbf{W}_{:,j \in \text{rows}[i]})$ 
7:    $\mathbf{B}_2 \leftarrow \text{BINARY}(\mathbf{W}_{:,j \notin \text{rows}[i]})$ 
8:   if  $\|\mathbf{W} - (\mathbf{B}_1 \cup \mathbf{B}_2)\|^2 < e$  then
9:      $e \leftarrow \|\mathbf{W} - (\mathbf{B}_1 \cup \mathbf{B}_2)\|^2$ 
10:     $K \leftarrow i$ 
11:   end if
12: end for
13: return  $\text{rows}[K]$ 

```

func HAARQUANT(\mathbf{W} , mode $\in \{\text{COL}, \text{ROW}\}$)

```

1:  $\mathbf{W}_{\text{low}} \leftarrow \mathcal{H}_{\text{low}}(\mathbf{W})$ 
2:  $p_1^* \leftarrow \text{SEG\_ROW\_SEARCH}(\mathbf{W}_{\text{low}})$ 
3:  $\mathbf{B}_{\text{low}} \leftarrow \text{BINARY}(\mathbf{W}_{|w_{i,j}| \leq p_1^*, \text{low}}^{(b)}) +$ 
    $\text{BINARY}(\mathbf{W}_{|w_{i,j}| > p_1^*, \text{low}}^{(b)})$ 
4:  $\mathbf{W}_{\text{diff}} \leftarrow \mathbf{W} - \mathcal{H}^{-1}(\mathbf{B}_{\text{low}}, \mathbf{0})$ 
5:  $\mathbf{W}_{\text{high}} \leftarrow \mathcal{H}_{\text{high}}(\mathbf{W}_{\text{diff}})$ 
6:  $p_2^* \leftarrow \text{SEG\_ROW\_SEARCH}(\mathbf{W}_{\text{high}})$ 
7:  $\mathbf{B}_{\text{high}} \leftarrow \text{BINARY}(\mathbf{W}_{|w_{i,j}| \leq p_2^*, \text{high}}^{(b)}) +$ 
    $\text{BINARY}(\mathbf{W}_{|w_{i,j}| > p_2^*, \text{high}}^{(b)})$ 
8:  $\mathbf{B} \leftarrow \mathbf{B}_{\text{low}} + \mathbf{B}_{\text{high}}$ 
9: return  $\mathbf{B}$ 

```

func BINARY(\mathbf{W})

```

1:  $\mu \leftarrow \frac{1}{m} \sum_{j=1}^m W_{\cdot j}$  // row-wise mean
2:  $\widetilde{\mathbf{W}} \leftarrow \mathbf{W} - \mu$  // centered matrix
3:  $\alpha = \sqrt{\frac{\|\widetilde{\mathbf{W}}\|_2^2}{m}}$  // row-wise scale
4:  $\mathbf{B} \leftarrow \alpha \cdot \text{sign}(\widetilde{\mathbf{W}}) + \mu$ 
5: return  $\mathbf{B}$ 

```

func SEG_ROW_SEARCH(\mathbf{W})

```

1:  $n \leftarrow \text{number of rows in } \mathbf{W}$ 
2:  $e \leftarrow +\infty \times \mathbf{1}_{n \times 1}$  // row-wise error
3:  $p^* \leftarrow \mathbf{0}_{n \times 1}$ 
   // optimal break-point of each row
4: for  $\tau = 0.1, 0.2, \dots, 0.9$  do
5:    $p \leftarrow \tau \times \max(\text{abs}(\mathbf{W}).(\text{dim} = 1))$ 
6:    $\mathbf{B}_1 \leftarrow \text{BINARY}(\mathbf{W}_{|w_{i \in [n], \cdot}| \leq p})$ 
7:    $\mathbf{B}_2 \leftarrow \text{BINARY}(\mathbf{W}_{|w_{i \in [n], \cdot}| > p})$ 
8:   if  $\|\mathbf{W} - (\mathbf{B}_1 + \mathbf{B}_2)\|^2 < e$  then
9:      $e \leftarrow \|\mathbf{W} - (\mathbf{B}_1 + \mathbf{B}_2)\|^2$ 
10:     $p^* \leftarrow p$ 
11:   end if
12: end for
13: return  $p^*$ 

```

331 Then, the average bit-width becomes:

$$\text{AvgBit} = \frac{\mathcal{M}_{\text{HBLLM-col}}}{nm} = 2 + \frac{112n \lceil \frac{m}{k} \rceil}{nm} + \frac{\tilde{m}}{nm} \approx 2.883 \text{ bits.} \quad (50)$$

332 **E HBLLM Implementation**

333 The implementation details of the HBLLM quantization pipeline are provided in Algorithm E.1.

334 **F Additional Experimental Results**

335 **F.1 Experimental Results for DeepSeek-R1-Distill-Llama-8B**

336 Table F.1 provides further experimental results on DeepSeek-R1-Distill-Llama-8B model. In line
 337 with the trends observed in Table 1, HBLLM consistently outperforms existing 1-2 bit quantization
 338 techniques across all these evaluation metrics. Remarkably, even when applied to the more complex
 339 and modern DeepSeek-R1-Distill-Llama-8B architecture, HBLLM preserves the same performance
 340 advantages previously demonstrated on LLaMA3-8B. This assessment highlights the effectiveness of
 341 HBLLM for LLMs.

342 **F.2 Comparison on 9 zero-shot QA datasets**

343 In the main paper, we report the average accuracy (AvgQA) across 9 zero-shot QA datasets to provide
 344 a high-level comparison of different quantization methods. In this appendix, we present the detailed

Table F.1: Perplexity and zero-shot accuracy results of DeepSeek-R1-Distill-Llama-8B.

Model	Method	Wbits	Perplexity↓			AvgQA↑
			C4	Wiki2	PTB	
DeepSeek-R1-Distill-Llama-8B	FullPrecision	16.00	18.40	13.14	20.57	63.80
	FrameQuant	2.20	59.60	46.71	70.79	43.95
	PB-LLM	1.70	316.3	224.6	344.3	34.52
	BiLLM	1.06	234.4	219.5	442.9	35.91
	ARB-LLM _X	1.06	74.31	54.73	69.35	42.92
	ARB-LLM _{RC}	1.06	67.77	54.27	92.37	43.00
	HBLLM-row	1.05	40.88	29.26	45.00	47.06
	HBLLM-col	1.00	55.82	35.80	62.80	45.71

Table F.2: Accuracy of 9 QA datasets on LLaMA1 family models. We compare the results among FrameQuant, PB-LLM, BiLLM, ARB-LLM and HBLLM to validate the quantization effect.

LLaMA1			Zero-shot Accuracy↑									AvgQA↑
Size	Method	Wbits	PIQA	BoolQ	OBQA	WinoG	ARC-e	ARC-c	HSwag	COPA	LAMBDA	
7B	FullPrecision	16.00	78.67	75.02	34.20	70.01	75.34	41.89	56.94	85.00	73.51	65.62
	FrameQuant	2.20	71.16	69.69	23.60	66.54	61.57	29.78	43.67	81.00	58.70	56.19
	PB-LLM	1.70	54.62	58.04	13.20	49.17	29.38	21.25	27.61	61.00	7.10	35.71
	BiLLM	1.09	59.63	54.62	15.00	53.20	35.27	19.80	30.38	71.00	21.15	40.01
	ARB-LLM _X	1.09	63.55	64.10	16.80	56.75	42.21	21.93	34.21	73.00	38.27	45.65
	ARB-LLM _{RC}	1.09	69.15	64.10	22.40	61.25	52.90	25.26	39.26	78.00	57.71	52.23
	HBLLM-row	1.08	73.67	71.07	24.60	62.75	64.81	30.80	47.60	79.00	63.03	57.48
	HBLLM-col	1.00	71.93	62.60	22.00	62.27	61.15	28.58	44.68	78.00	55.09	54.03
13B	FullPrecision	16.00	79.16	77.92	33.20	72.61	77.36	46.42	59.93	90.00	76.19	68.09
	FrameQuant	2.20	75.14	73.64	26.20	68.67	66.71	34.64	48.58	83.00	69.59	60.69
	PB-LLM	1.70	58.00	62.02	14.60	52.96	33.25	18.26	29.99	69.00	25.44	40.39
	BiLLM	1.10	67.19	67.09	19.00	60.69	53.66	25.34	39.90	74.00	51.12	50.89
	ARB-LLM _X	1.10	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	ARB-LLM _{RC}	1.10	72.03	72.51	24.40	68.11	64.73	31.14	45.54	88.00	69.80	59.58
	HBLLM-row	1.09	76.28	68.93	28.60	68.90	70.37	38.31	52.54	86.00	73.22	62.57
	HBLLM-col	1.00	75.03	70.46	24.80	69.69	69.02	34.39	51.01	87.00	69.82	61.25
30B	FullPrecision	16.00	80.96	82.69	36.00	75.93	80.35	52.73	63.35	90.00	77.55	71.06
	FrameQuant	2.20	76.39	73.58	31.20	72.85	72.18	38.14	53.97	92.00	75.84	65.13
	PB-LLM	1.70	63.60	64.34	16.00	60.30	44.78	20.90	34.80	74.00	46.30	47.22
	BiLLM	1.11	71.49	68.87	24.40	67.96	63.59	29.69	44.51	84.00	68.10	58.07
	ARB-LLM _X	1.11	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	ARB-LLM _{RC}	1.11	74.86	76.64	29.60	73.64	70.62	37.46	50.29	91.00	76.31	64.49
	HBLLM-row	1.10	77.09	79.72	32.20	71.67	74.24	43.34	55.79	91.00	75.80	66.76
	HBLLM-col	1.00	76.66	72.23	29.80	71.35	73.91	39.68	53.81	90.00	76.29	64.86
65B	FullPrecision	16.00	81.34	84.86	38.00	77.43	81.31	52.82	64.56	91.00	79.12	72.27
	FrameQuant	2.20	79.00	83.27	32.00	74.66	77.23	45.56	56.58	90.00	78.92	68.58
	PB-LLM	1.70	71.98	79.45	28.20	74.98	67.34	34.47	46.57	89.00	70.37	62.48
	BiLLM	1.10	74.05	80.40	26.20	70.40	69.32	36.60	48.15	85.00	68.35	62.05
	ARB-LLM _X	1.10	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	ARB-LLM _{RC}	1.10	77.97	83.94	32.60	75.93	76.64	44.28	55.58	90.00	79.84	68.53
	HBLLM-row	1.09	78.89	81.96	33.80	75.77	76.94	45.65	59.03	91.00	79.58	69.18
	HBLLM-col	1.00	79.00	81.47	31.40	74.51	75.17	42.66	57.98	91.00	77.28	67.83

accuracy on each individual dataset. As shown in Tables F.2–F.5 our method consistently delivers strong performance across all datasets, further validating its effectiveness and robustness in diverse zero-shot question answering tasks.

F.3 Comparison of Avg. Relative Perplexity and Avg. Relative QA Accuracy across Models

To provide a unified view of model performance across both language modeling and common sense reasoning tasks, we compute two metrics: **Avg. Relative Perplexity** and **Avg. Relative QA**. **Avg. Relative Perplexity** is calculated as the average over three language modeling datasets: C4, Wiki2 and PTB. **Avg. Relative QA** is computed as the average across 9 zero-shot QA datasets.

Table F.3: Accuracy of 9 QA datasets on LLaMA2 family models. We compare the results among FrameQuant, PB-LLM, BiLLM, ARB-LLM and HBLLM to validate the quantization effect.

LLaMA2			Zero-shot Accuracy↑									AvgQA↑
Size	Method	Wbits	PIQA	BoolQ	OBQA	WinoG	ARC-e	ARC-c	HSwag	COPA	LAMBD	
7B	FullPrecision	16.00	76.44	79.72	33.40	66.46	73.91	44.20	57.80	87.00	70.95	65.54
	FrameQuant	2.20	68.55	67.13	19.00	61.80	56.61	27.56	42.20	78.00	53.89	52.75
	PB-LLM	1.70	54.84	61.90	11.80	48.07	27.90	19.45	27.78	60.00	17.14	36.54
	BiLLM	1.08	59.85	63.82	12.80	54.14	40.15	21.67	31.66	64.00	30.93	42.11
	ARB-LLM _X	1.08	61.97	66.42	16.20	57.70	43.27	22.78	33.81	68.00	38.54	45.41
	ARB-LLM _{RC}	1.08	61.15	59.51	18.00	59.27	41.75	23.46	39.62	67.00	50.61	46.71
	HBLLM-row	1.07	72.96	73.58	25.60	60.93	61.87	32.17	47.50	84.00	61.03	57.74
	HBLLM-col	1.00	71.11	69.66	21.40	61.33	59.22	29.35	45.33	74.00	55.44	54.09
13B	FullPrecision	16.00	79.05	80.55	35.20	72.14	79.42	48.46	60.05	91.00	76.73	69.18
	FrameQuant	2.20	73.83	76.30	25.60	69.61	69.02	33.79	47.15	85.00	71.84	61.35
	PB-LLM	1.70	54.30	41.25	12.60	50.12	27.27	20.14	27.00	59.00	4.52	32.91
	BiLLM	1.08	61.81	66.51	18.00	56.35	43.64	21.84	33.33	75.00	44.34	46.76
	ARB-LLM _X	1.08	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	ARB-LLM _{RC}	1.08	69.64	72.42	25.40	64.17	62.12	30.03	40.99	85.00	66.35	57.35
	HBLLM-row	1.07	75.63	77.74	29.80	68.27	71.21	37.12	52.54	88.00	72.19	63.61
	HBLLM-col	1.00	74.86	77.61	26.80	69.14	70.66	35.75	50.46	84.00	69.12	62.04
70B	FullPrecision	16.00	82.26	83.76	37.20	77.98	82.74	54.35	64.77	94.00	79.60	72.96
	FrameQuant	2.20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	PB-LLM	1.70	64.04	74.77	21.20	64.72	55.64	27.22	38.77	80.00	61.94	54.26
	BiLLM	1.09	68.39	70.64	24.40	65.98	64.23	32.34	41.85	82.00	52.47	55.81
	ARB-LLM _X	1.09	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	ARB-LLM _{RC}	1.09	77.53	80.21	34.40	75.85	77.31	44.80	55.84	92.00	80.98	68.77
	HBLLM-row	1.08	79.65	81.56	33.40	75.06	79.29	50.51	59.02	91.00	80.56	70.01
	HBLLM-col	1.00	78.84	78.99	30.80	76.24	77.99	45.39	58.61	92.00	78.59	68.61

Table F.4: Accuracy of 9 QA datasets on LLaMA3 family models. We compare the results among FrameQuant, PB-LLM, BiLLM, ARB-LLM and HBLLM to validate the quantization effect.

LLaMA3			Zero-shot Accuracy↑									AvgQA↑
Size	Method	Wbits	PIQA	BoolQ	OBQA	WinoG	ARC-e	ARC-c	HSwag	COPA	LAMBD	
8B	FullPrecision	16.00	78.35	83.18	34.20	71.67	81.61	52.90	57.66	89.00	71.88	68.94
	FrameQuant	2.20	65.40	71.87	19.20	59.04	60.19	29.69	47.06	78.00	46.94	52.27
	PB-LLM	1.70	57.18	62.26	11.60	50.36	31.31	18.00	28.59	55.00	17.16	36.83
	BiLLM	1.06	59.14	64.46	15.20	53.75	37.54	19.45	32.01	65.00	30.04	41.84
	ARB-LLM _X	1.06	61.15	63.91	15.00	56.43	45.37	20.14	31.84	66.00	36.07	43.40
	ARB-LLM _{RC}	1.06	62.73	69.72	18.80	56.91	50.34	25.94	35.18	74.00	49.08	49.08
	HBLLM-row	1.05	67.68	72.94	23.80	63.54	56.52	28.07	43.55	81.00	56.14	54.80
	HBLLM-col	1.00	67.03	69.94	21.20	60.93	49.79	25.85	42.79	81.00	44.30	51.43
70B	FullPrecision	16.00	82.15	85.38	37.80	80.51	86.87	60.07	66.34	93.00	79.45	74.62
	FrameQuant	2.20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	PB-LLM	1.70	57.89	68.07	17.60	58.96	37.37	19.62	38.86	80.00	48.67	47.45
	BiLLM	1.09	52.39	42.26	12.60	51.78	25.34	20.73	28.11	65.00	9.41	34.18
	ARB-LLM _X	1.09	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	ARB-LLM _{RC}	1.09	74.86	80.89	25.80	72.69	72.69	40.44	51.99	85.00	70.76	63.90
	HBLLM-row	1.08	52.88	83.18	28.80	78.06	27.19	20.48	53.83	89.00	74.64	56.45
	HBLLM-col	1.00	56.04	77.83	29.00	73.40	33.63	18.34	53.19	91.00	70.56	55.89

Table F.5: Accuracy of 9 QA datasets on DeepSeek-R1-Distill-Llama-8B. We compare the results among FrameQuant, PB-LLM, BiLLM, ARB-LLM and HBLLM to validate the quantization effect.

DeepSeek-R1-Distill-Llama			Zero-shot Accuracy↑									AvgQA↑
Size	Method	Wbits	PIQA	BoolQ	OBQA	WinoG	ARC-e	ARC-c	HSwag	COPA	LAMBD	
8B	FullPrecision	16.00	76.33	82.94	31.40	67.48	70.54	40.27	55.54	89.00	60.72	63.80
	FrameQuant	2.20	61.48	65.50	17.00	55.17	42.05	22.10	35.42	67.00	29.81	43.95
	PB-LLM	1.70	55.01	59.54	12.60	48.07	29.08	16.98	27.56	52.00	9.82	34.52
	BiLLM	1.06	55.01	60.37	12.80	49.41	26.89	19.62	29.20	57.00	12.87	35.91
	ARB-LLM _X	1.06	60.34	67.55	15.60	55.64	39.90	22.35	32.97	68.00	23.97	42.92
	ARB-LLM _{RC}	1.06	60.50	63.15	15.40	53.51	40.24	21.76	34.55	64.00	33.90	43.00
	HBLLM-row	1.05	63.71	72.26	18.00	56.20	41.75	23.38	39.74	74.00	34.47	47.06
	HBLLM-col	1.00	64.53	68.53	18.40	56.35	42.42	22.44	39.75	70.00	28.99	45.71

Table F.6: Accuracy of 9 QA datasets on OPT family models. We compare the results among FrameQuant, PB-LLM, BiLLM, ARB-LLM and HBLLM to validate the quantization effect.

OPT			Zero-shot Accuracy↑									AvgQA↑
Size	Method	Wbits	PIQA	BoolQ	OBQA	WinoG	ARC-e	ARC-c	HSwag	COPA	LAMB	
1.3B	FullPrecision	16.00	71.65	57.77	23.40	59.27	57.03	23.38	41.53	81.00	57.85	52.54
	FrameQuant	2.20	64.91	57.00	16.40	55.17	46.51	20.73	34.01	71.00	34.58	44.48
	PB-LLM	1.70	53.81	48.38	12.60	50.67	28.49	19.97	26.20	59.00	1.88	33.44
	BiLLM	1.09	59.41	61.07	13.80	52.49	35.65	17.06	29.53	62.00	14.54	38.39
	ARB-LLM _X	1.09	60.39	60.61	14.60	52.72	39.98	18.09	30.65	64.00	31.73	41.42
	ARB-LLM _{RC}	1.09	65.13	56.88	17.60	53.75	47.22	20.05	33.58	71.00	42.27	45.28
	HBLLM-row	1.07	66.49	62.14	17.80	56.20	47.94	21.42	35.70	69.00	40.42	46.35
	HBLLM-col	1.00	65.61	50.18	18.20	56.43	45.75	20.90	35.25	72.00	38.02	44.70
2.7B	FullPrecision	16.00	73.83	60.34	25.00	61.33	60.73	26.79	45.88	77.00	63.63	54.95
	FrameQuant	2.20	67.36	61.93	18.20	57.22	52.23	22.70	36.88	75.00	54.71	49.58
	PB-LLM	1.70	54.79	62.11	13.00	50.67	30.13	19.20	27.35	69.00	12.30	37.62
	BiLLM	1.10	60.45	62.08	13.20	53.59	35.90	20.56	30.54	63.00	20.88	40.02
	ARB-LLM _X	1.10	63.00	62.35	15.40	54.78	42.85	19.45	32.20	69.00	42.38	44.60
	ARB-LLM _{RC}	1.10	67.74	57.03	18.20	58.17	51.05	22.61	37.33	74.00	59.62	49.53
	HBLLM-row	1.09	68.66	58.35	20.20	56.91	52.86	22.87	39.49	72.00	47.86	48.80
	HBLLM-col	1.00	67.79	62.51	19.40	56.27	51.05	22.18	37.24	76.00	44.61	48.56
13B	FullPrecision	16.00	75.84	65.75	27.00	65.19	67.09	32.94	52.45	81.00	68.66	58.41
	FrameQuant	2.20	72.85	66.15	22.60	63.69	62.54	28.07	45.49	82.00	68.25	55.42
	PB-LLM	1.70	54.90	62.17	12.80	52.01	29.46	20.99	26.68	57.00	9.96	39.50
	BiLLM	1.13	67.25	63.91	18.20	58.64	51.73	24.66	38.18	76.00	54.03	49.82
	ARB-LLM _X	1.13	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	ARB-LLM _{RC}	1.13	73.34	67.92	23.40	64.25	60.14	27.30	44.41	82.00	67.44	55.35
	HBLLM-row	1.12	73.94	67.98	22.60	61.64	62.50	29.10	46.55	83.00	66.64	55.91
	HBLLM-col	1.00	73.56	66.51	22.00	64.56	62.04	28.92	45.65	82.00	68.06	55.66
30B	FullPrecision	16.00	77.58	70.40	30.20	68.35	70.03	34.47	54.28	82.00	71.47	62.09
	FrameQuant	2.20	75.08	70.49	26.60	64.48	66.79	30.72	48.37	82.00	72.04	59.62
	PB-LLM	1.70	64.47	62.94	16.40	53.67	44.65	21.84	35.95	70.00	45.37	46.14
	BiLLM	1.13	71.33	63.91	21.80	61.80	57.87	25.09	41.94	80.00	64.23	54.22
	ARB-LLM _X	1.13	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	ARB-LLM _{RC}	1.13	74.27	67.37	27.40	64.33	63.76	30.03	48.03	78.00	74.15	58.59
	HBLLM-row	1.12	75.46	70.80	26.80	66.22	66.12	30.97	49.76	82.00	72.23	60.04
	HBLLM-col	1.00	74.70	68.90	25.00	64.96	66.16	30.03	48.90	81.00	70.52	58.91

353 **Relative Perplexity** is defined as:

$$RS_{\text{Perplexity}} := \frac{S_{\text{Perplexity}}}{S_{\text{Perplexity}}^{\text{FP}}}, \quad (51)$$

354 where $S_{\text{Perplexity}}$ and $S_{\text{Perplexity}}^{\text{FP}}$ are the perplexities of the models after and before quantization, respec-
355 tively.

356 **Relative QA** is defined as:

$$RS_{\text{QA}} := \frac{S_{\text{QA}}}{S_{\text{QA}}^{\text{FP}}}. \quad (52)$$

357 where S_{QA} and $S_{\text{QA}}^{\text{FP}}$ are the QA scores of the models after and before quantization, respectively.

358 Figure F.1 presents the comparison of these two metrics across the LLaMA-1, LLaMA-2, LLaMA-3,
359 and OPT family models. HBLLM consistently shows lower relative perplexity and higher relative
360 QA accuracy compared to prior 1–2 bit quantization methods (PB-LLM, BiLLM, and ARB-LLM_{RC}),
361 demonstrating both effectiveness and scalability across diverse LLMs.

362 F.4 Ablation Study from the Perspective of ℓ_2 Relative Error

363 To complement the ablation studies presented in the main paper, we provide an additional analysis
364 from the perspective of relative quantization error in terms of the Frobenius norm. Specifically, we
365 compute the relative ℓ_2 error for each quantized weight block, defined as the Frobenius norm of the
366 quantization residual normalized by the original weight norm. We then examine the distribution of
367 these errors across different layers and quantization methods to assess fidelity at the matrix level.

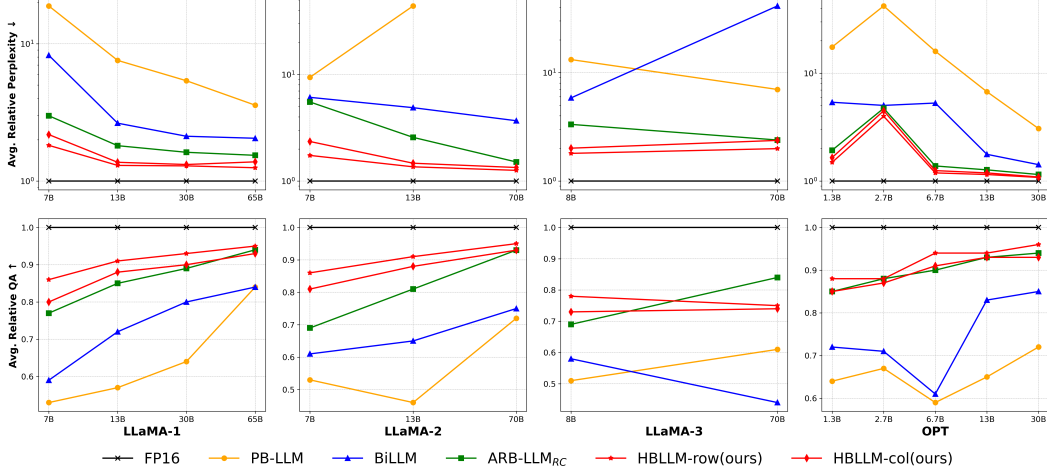


Figure F.1: Average relative perplexity and average relative QA accuracy (normalized to FP16) for LLaMA-1/2/3 family models, comparing LLM binarization methods and our HBLLM.

368 The relative error of a quantized weight matrix is defined as:

$$RE_{\text{weight}} = \frac{\|W_{\text{FP}} - W_{\text{B}}\|_F^2}{\|W_{\text{FP}}\|_F^2}. \quad (53)$$

369 where W_{FP} and W_{B} denote the full-precision and quantized weight matrices, respectively.

370 Figure F.2 visualizes the sorted relative quantization errors under different ablation settings, including
 371 (a) salient column selection criterion, (b) grouping granularity, (c) shared mean strategy, and (d)
 372 number of partitioning candidates. Across all cases, our proposed HBLLM consistently achieves
 373 lower median relative error compared to prior approaches, further confirming the effectiveness of
 374 each component.

375 In Figure F.2a, we observe that using the ℓ_2 norm as the column selection criterion leads to significantly
 376 lower quantization errors than ℓ_1 , validating its stronger ability to capture the energy distribution of
 377 weights. In Figure F.2b, row-wise grouping substantially outperforms global grouping, indicating
 378 the importance of fine-grained adaptivity. In Figure F.2c, the shared mean strategy slightly reduces
 379 quantization error while offering better compression efficiency. Finally, in Figure F.2d, increasing
 380 the number of partition candidates reduces error up to a point, with 40 candidates offering the best
 381 trade-off.

382 These observations align well with perplexity-based trends, and jointly confirm the robustness and
 383 precision of HBLLM under various quantization design choices.

384 G Inference Latency Estimation and Efficiency Analysis

385 To evaluate the computational efficiency of HBLLM in practical inference scenarios, we design an
 386 estimation-based experiment to analyze its inference latency. This is necessary because no current
 387 inference framework fully supports the dequantization algorithm used in HBLLM, and constructing
 388 such a backend involves significant engineering effort beyond the current scope.

389 Our findings suggest that the inference latency of HBLLM is approximately 31.8% of the FP16
 390 baseline, indicating strong computational benefits despite the algorithm’s structural complexity.

391 **Latency Estimation Methodology.** We define the relative inference time of HBLLM compared to
 392 FP16 as:

$$R(p, l) := \frac{T_{\text{HBLLM}}}{T_{\text{FP16}}} = (1 - p) + \frac{p}{l}, \quad (54)$$

393 where p is the portion of time spent on matrix-vector multiplication (GEMV) during inference, and l
 394 is the acceleration factor of GEMV after quantization. Following prior works such as GPTQ [?], we
 395 use $p = 0.78$.

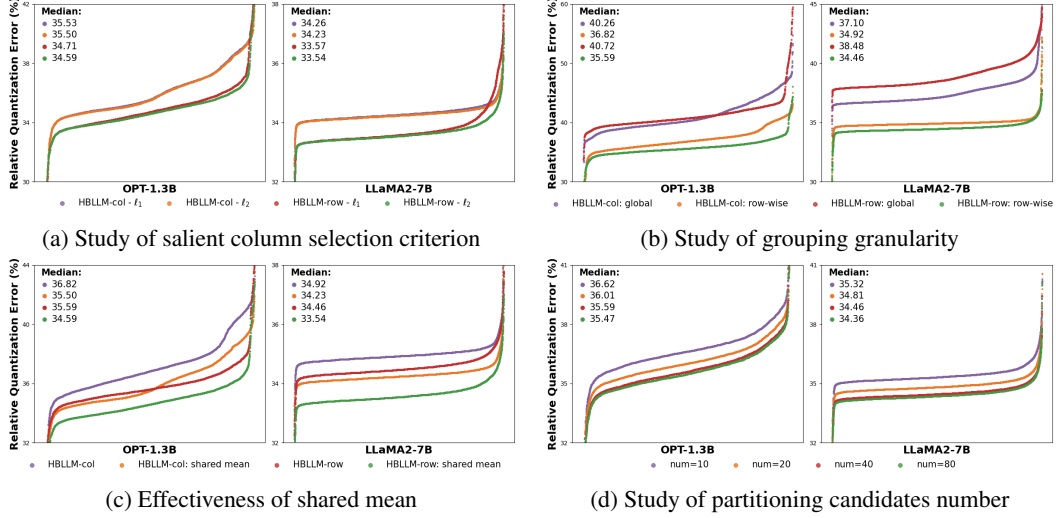


Figure F.2: Ablation study on OPT-1.3B and LLaMA2-7B. Results are measured by ℓ_2 Relative Quantization Error.

To compute l , we measure:

$$l := \frac{T_{\text{torch}}}{T_{\text{hqmV}}}, \quad (55)$$

where T_{torch} denotes the runtime of FP16 GEMV using PyTorch, and T_{hqmV} refers to our quantized matrix-vector multiplication kernel (HQM).

Since the current hqmV implementation does not support Intra-Frequency Grouping (IFG), we use the runtime of HBLLM-col without IFG (denoted HBLLM-col w/o IFG) as a proxy to estimate the runtime of HBLLM-col with full IFG. Notably, applying IFG doubles the number of groups, potentially leading to: roughly $2\times$ more CUDA warp divergence, a 90% increase in average data loading (Avgbit increases to 2.88 bit).

However, since data loading and computation can overlap on GPU, we conservatively estimate that the runtime with IFG should not exceed twice the runtime without IFG.

Table G.1: GEMV Runtime and Inference Latency Estimation for HBLLM (OPT-175B Linear Layer, P100 GPU).

Method	T_{torch} (s)	T_{hqmV} (s)	$R(p, l)$
FP16 Baseline	1.35×10^{-3}	—	1.00
HBLLM-col (w/o IFG)	—	8.54×10^{-5}	—
HBLLM-col (with IFG)	—	1.70×10^{-4}	0.318

• **Setup:** We benchmark a single linear layer of OPT-175B under the same GEMV input/output shape used in GPTQ [12]. All timing experiments are conducted on an NVIDIA P100 GPU.

- Measured: $T_{\text{torch}} = 1.35 \times 10^{-3}$ s
- Measured: T_{hqmV} for HBLLM-col w/o IFG = 8.54×10^{-5} s
- Estimated: T_{hqmV} for full HBLLM-col (with IFG) $\approx 1.70 \times 10^{-4}$ s

This yields:

$$l = \frac{1.35 \times 10^{-3}}{1.70 \times 10^{-4}} = 7.94, \quad R = (1 - 0.78) + \frac{0.78}{7.94} \approx \boxed{0.318} \quad (56)$$

• **Discussion:** This result indicates that HBLLM inference, despite involving additional processing (e.g., dequantization and grouping), maintains high efficiency due to:

- Lightweight binary matrix operations;
- Reduced memory bandwidth consumption;
- Effective overlap of memory access and computation in GPU execution.

Overall, HBLLM achieves strong acceleration over FP16 without relying on highly specialized hardware or handcrafted fusion kernels, making it a viable candidate for practical deployment in low-bit LLM inference systems.

H Licenses for Existing Assets

[32], llama, llama2, llama3
<https://huggingface.co/TheBloke/llama, llama2, llama3>
 [37], <https://github.com/facebookresearch/metaseq/tree/main/projects/OPT>, OPT-175B LICENSE AGREEMENT
 [15], <https://github.com/Aaronhuang-778/BiLLM>, MIT License
 [18], <https://github.com/ZHITENGLI/ARB-LLM>, Apache License 2.0
 [1], <https://github.com/vsingh-group/FrameQuant>
 [29], <https://github.com/scotfree/PbLLM>, Creative Commons Zero v1.0 Universal
 [4], <https://leaderboard.allenai.org/physicaliqa/submissions/get-started>
 [7], <https://github.com/google-research-datasets/boolean-questions>
 [8], https://huggingface.co/datasets/allenai/ai2_arc, Creative Commons Attribution Share Alike 4.0
 [21], https://huggingface.co/datasets/ptb-text-only/ptb_text_only, Dataset provided for research purposes only
 [22], <https://huggingface.co/datasets/mindchain/wikitext2>, Creative Commons Attribution-ShareAlike License.
 [23], <https://github.com/allenai/OpenBookQA>, Apache License 2.0
 [24], <https://github.com/EleutherAI/lm-evaluation-harness>, MIT License
 [25], <https://zenodo.org/records/2630551>, Creative Commons Attribution 4.0 International
 [26], <https://github.com/google-research/text-to-text-transfer-transformer#datasets>, Apache License 2.0
 [27], <https://asgordon.github.io/copa.html>, BSD 2-Clause License
 [28], <https://github.com/allenai/winogrande>, Apache License 2.0

References

- [1] Adepu H, Zeng Z, Zhang L, et al. FrameQuant: flexible low-bit quantization for transformers. *Proceedings of the 41st International Conference on Machine Learning*. 2024: 203–227.
- [2] Ashkboos S, Croci M L, do Nascimento M G, et al. SliceGPT: Compress Large Language Models by Deleting Rows and Columns. *The Twelfth International Conference on Learning Representations*. 2024.
- [3] Lin J, Tang J, Tang H, et al. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. *Proceedings of Machine Learning and Systems*, 2024, 6: 87–100.
- [4] Bisk Y, Zellers R, Gao J, et al. PIQA: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020: 7432–7439.
- [5] Chee J, Cai Y, Kuleshov V, et al. QuIP: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 2023, 36: 4396–4429.
- [6] Chen H, Lv C, Ding L, et al. DB-LLM: Accurate dual-binarization for efficient LLMs. *Findings of the Association for Computational Linguistics: ACL 2024*. 2024: 8719–8730.
- [7] Clark C, Lee K, Chang M W, et al. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *Proceedings of NAACL-HLT*. 2019: 2924–2936.
- [8] Clark P, Cowhey I, Etzioni O, et al. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- [9] Duan Y, Liu F, Jiao L, et al. SAR image segmentation based on convolutional-wavelet neural network and Markov random field. *Pattern Recognition*, 2017, 64: 255–267.
- [10] Edalati A, Ghaffari A, Nejad M G, et al. OAC: Output-adaptive calibration for accurate post-training quantization. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2025, 39(16): 16453–16461.
- [11] Frantar E, Alistarh D. SparseGPT: Massive language models can be accurately pruned in one-shot. *International Conference on Machine Learning*. 2023: 10323–10337.
- [12] Frantar E, Ashkboos S, Hoefler T, et al. OPTQ: Accurate post-training quantization for generative pre-trained transformers. *11th International Conference on Learning Representations*. 2023.

- [13] Gong R, Ding Y, Wang Z, et al. A survey of low-bit large language models: Basics, systems, and algorithms. *arXiv preprint arXiv:2409.16694*, 2024.
- [14] Huang H, He R, Sun Z, et al. Wavelet-SRNet: A wavelet-based CNN for multi-scale face super resolution. *Proceedings of the IEEE International Conference on Computer Vision*. 2017: 1689–1697.
- [15] Huang W, Liu Y, Qin H, et al. BiLLM: Pushing the limit of post-training quantization for LLMs. *Proceedings of the 41st International Conference on Machine Learning*. 2024: 20023–20042.
- [16] Huang W, Zheng X, Ma X, et al. An empirical study of LLaMA3 quantization: From LLMs to MLLMs. *Visual Intelligence*, 2024, 2(1): 36.
- [17] Jo D, Kim T, Kim Y. Mixture of scales: Memory-efficient token-adaptive binarization for large language models. *Advances in Neural Information Processing Systems*, 2024, 37: 137474–137494.
- [18] Li Z, Yan X, Zhang T, et al. Arb-LLM: Alternating refined binarizations for large language models. *arXiv preprint arXiv:2410.03129*, 2024.
- [19] Liang C, Zuo S, Zhang Q, et al. Less is more: Task-aware layer-wise distillation for language model compression. *International Conference on Machine Learning*. 2023: 20852–20867.
- [20] Mallat S. *A Wavelet Tour of Signal Processing*. Elsevier, 1999.
- [21] Marcus M, Kim G, Marcinkiewicz M A, et al. The Penn Treebank: Annotating predicate argument structure. *Human Language Technology: Proceedings of a Workshop held at Plainsboro*. 1994.
- [22] Merity S, Xiong C, Bradbury J, et al. Pointer Sentinel Mixture Models. *International Conference on Learning Representations*. 2017.
- [23] Mihaylov T, Clark P, Khot T, et al. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018: 2381–2391.
- [24] Mukherjee S, Liu X, Zheng G, et al. Few-shot learning evaluation in natural language understanding. *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2021.
- [25] Paperno D, Kruszewski G, Lazaridou A, et al. The LAMBADA dataset: Word prediction requiring a broad discourse context. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 2016: 1525–1534.
- [26] Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020, 21(140): 1–67.
- [27] Roemmele M, Bejan C A, Gordon A S. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*. 2011: 90–95.
- [28] Sakaguchi K, Bras R L, Bhagavatula C, et al. Winogrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 2021, 64(9): 99–106.
- [29] Shang Y, Yuan Z, Wu Q, et al. PB-LLM: Partially binarized large language models. *The Twelfth International Conference on Learning Representations*. 2024.
- [30] Shridhar K, Stolfo A, Sachan M. Distilling reasoning capabilities into smaller language models. *Findings of the Association for Computational Linguistics: ACL 2023*, 2023: 7059–7073.
- [31] Sun M, Liu Z, Bair A, et al. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.
- [32] Touvron H, Lavril T, Izacard G, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [33] Xiao G, Lin J, Seznec M, et al. SmoothQuant: Accurate and efficient post-training quantization for large language models. *International Conference on Machine Learning*. 2023: 38087–38099.
- [34] Xu Y, Han X, Yang Z, et al. OneBit: Towards extremely low-bit large language models. *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024: 66357–66382.
- [35] Yao Z, Yazdani Aminabadi R, Zhang M, et al. ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 2022, 35: 27168–27183.
- [36] Zellers R, Holtzman A, Bisk Y, et al. HellaSwag: Can a machine really finish your sentence? *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019: 4791–4800.
- [37] Zhang S, Roller S, Goyal N, et al. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.