MBVQ: MULTI-LEVEL BLOCK-WISE VECTOR QUANTIZATION FOR HARDWARE FRIENDLY POST-TRAINING QUANTIZATION OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

The rapid scaling of large language models (LLMs) has substantially increased their deployment cost. Post-training quantization (PTQ), especially vector quantization (VQ), focuses on model compression to below 2 bits per parameter, but usually overlooks the data movement bottleneck that dominates real-world inference latency. In this paper, we propose **M**ulti-level **B**lock-wise **V**ector **Q**uantization (MBVQ), a hardware-friendly VQ framework designed to address this issue. MBVQ introduces two key innovations: (1) capacity-constrained per-block codebooks for efficient memory residency during dequantization, and (2) multi-level residual quantization scheme with row-wise importance selection and per-block scaling to compensate for reconstruction error. Specifically, we first partition weight matrices into blocks to adapt large-scale codebooks to varying hardwares. Subsequently, we merge redundant adjacent blocks for quantizing residuals with selective codebooks and design codebook rescaling to adaptively adjust the block-wise distribution of values to simultaneously improve model accuracy and achieve substantial inference speedup. Experiments on a range of LLMs demonstrate that MBVQ consistently outperforms state-of-the-art low-bit quantization baselines under comparable compression ratios, delivering approximately $6.62\times$ inference speedup, while maintaining competitive perplexity and zero-shot accuracy.

1 INTRODUCTION

Large language models (LLMs) have undergone rapid advancements in recent years. Open-source models such as LLaMA (Touvron et al., 2023a;b; Grattafiori et al., 2024) and Qwen (Bai et al., 2023; Yang et al., 2025), in particular, have attracted widespread attention due to their transparency and remarkable performance. However, the growing scale of these models comes at a substantial cost: their enormous architectural footprint poses significant challenges for real-world deployment and practical application (Chen et al., 2026). To address this, researchers have turned to model-level compression techniques—including quantization, pruning, knowledge distillation, approximation, and parameter sharing—to reduce model size while preserving performance and enabling hardware-efficient inference (Lee et al., 2026).

Post-training quantization (PTQ) reduces high-precision pretrained model to low-bit representations and has been extensively studied in recent years owing to its simplicity, low cost, and ability of rapid deployment (Nagel et al., 2021; Gholami et al., 2022). GPTQ (Frantar et al., 2022) leverages the Hessian matrix to compensate for errors on more important weights. AWQ (Lin et al., 2024) employs activation-aware channel-wise scaling. However, since scalar quantization ignores the structural similarities inherent in weight matrices, these methods incur non-negligible performance degradation under extreme quantization regimes (2-bit and below) (Chee et al., 2023; Tseng et al., 2024a; Egiazarian et al., 2024). Consequently, vector quantization (VQ) approaches gain prominence in the ultra-low-bit compression landscape. Notably, AQLM (Egiazarian et al., 2024) achieves practical model accuracy at an average of 2 bits per parameter, while CRVQ (Xu et al., 2025) substantially improves performance under the extreme 1-bit setting. However, these methods focus on pursuing model fidelity under aggressive compression, but usually overlook the critical obstacle to real-world deployment.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Recently, the primary bottleneck in LLM inference has been shifting from computation to data movement dominated by input and output bandwidth (Chowdhury, 2026). Existing VQ methods design codebooks that focus on optimizing the trade-off between model accuracy and compression ratio, and neglect the fundamental hardware friendly issue due to the fact that large codebooks cannot reside in on-chip memory. During dequantization, an oversized codebook with low local hit rates triggers frequent codebook page swaps, and incurs substantial latency in memory access.

To address the problem, in this paper, we propose **M**ulti-level **B**lock-wise **V**ector **Q**uantization (MBVQ) that is explicitly designed to be memory-access-friendly. Specifically, we first partition each weight matrix into fixed-size blocks, each assigned a dedicated, capacity-constrained small codebook. We then introduce a block merging mechanism to eliminate codebook redundancy across adjacent blocks. Finally, we employ multi-level residual codebook quantization with learnable per-block scaling to further recover model accuracy. Our contributions are summarized below.

- We propose a hardware friendly vector quantization framework for post-training quantization that partitions the entire weight matrix into blocks for designing capacity-constrained block-wise codebooks and accommodating to diverse hardware deployment requirements.
- We develop a block merging strategy that further compresses the number of codebooks beyond block-wise quantization to simultaneously improve model accuracy and achieve substantial inference acceleration.
- We design a codebook rescaling scheme that adjusts the distribution of values within each block to a well-conditioned range to compensate for block-wise vector quantization.

2 Related Work

Quantization is an effective technique for compressing and accelerating large language models (Gholami et al., 2022; Nagel et al., 2021). Depending on when quantization is applied, existing methods can be broadly categorized into quantization-aware training (QAT) and post-training quantization (PTQ). QAT simulates the quantization loss during model training to compensate for the degradation introduced by reduced precision (Liu et al., 2024b; Dettmers et al., 2023; Xu et al., 2024; Jo et al., 2024), whereas PTQ directly quantizes a pretrained model, typically using a small set of calibration data to compensate for quantization error.

Most mainstream low-bit compression methods adopt the PTQ paradigm. Representative scalar quantization approaches include GPTQ (Frantar et al., 2022) and AWQ (Lin et al., 2024): the former performs layer-wise weight quantization using the Hessian matrix for error compensation, while the latter leverages activation distributions to identify and protect weights that are more critical to model output. More recent scalar quantization research has largely focused on mixed-precision strategies, such as AMQ (Lee et al., 2025), MLWQ (Hu et al., 2025), and FlexQuant (Liu et al., 2025a). However, due to the severe performance degradation of scalar quantization under extreme compression regimes, vector quantization (VQ) has become the predominant approach for sub-2-bit settings.

GPTVQ (van Baalen et al., 2025) co-designs VQ representation, PTQ flow, and a mobile inference engine, enabling efficient CPU LUT-based decoding with simultaneous footprint reduction and token rate improvement over INT4. VQ-LLM (Liu et al., 2025b) proposes a codebook cache and codebook-centric compute engine to generate high-performance fused VQ kernels on GPUs, translating VQ's memory savings into real latency gains. VPTQ (Liu et al., 2024a) introduces channel-independent second-order optimization with residual and outlier quantization. QuIP# (Tseng et al., 2024a) pushes VQ to the 2-bit frontier through randomized Hadamard transforms and the E8P lattice codebook. AQLM (Egiazarian et al., 2024) adapts additive quantization from the information retrieval literature and couples it with Transformer block-level joint optimization, enabling VQ to operate at even lower bit widths. Building on AQLM, CRVQ (Xu et al., 2025) achieves strong results at 1 bit by reordering salient channels and employing extended codebooks. RSAVQ (Xu et al., 2026) introduces Riemannian geometry to measure weight sensitivity, thereby allocating differentiated quantization precision across parameters. GLVQ (Zhang et al., 2026) and PCDVQ (Yue et al., 2025) explore non-conventional codebook designs for storing quantized weights.

The deployment bottleneck of quantized models has increasingly shifted toward data movement (Dao et al., 2022; Chowdhury, 2026; Gholami et al., 2022). In response, a growing number of

methods have begun to pursue relatively hardware-friendly VQ schemes. CRVQ (Xu et al., 2025) and SWSC (Zeng et al., 2025) both exploit the distributional properties of weights along channels, enabling channel-group-wise dequantization rather than matrix-wide dequantization. LoPRo (Gu et al., 2026) similarly employs salient-column reordering and block-wise rotation, avoiding the need to load the entire weight matrix into memory during dequantization.

Although the above methods differentiate channel importance and attempt channel grouping during quantization, they all rely on channel reordering operations, which are inherently memory-unfriendly. Moreover, their codebook sizes remain unconstrained and typically cannot reside in on-chip memory. To address these limitations, we propose a hardware-friendly block-wise vector quantization method with explicitly bounded codebook sizes. In MBVQ, each codebook is small enough to remain memory-resident during dequantization, substantially reducing I/O operations at inference time.

3 PRELIMINARIES

3.1 RESIDUAL VECTOR QUANTIZATION

Vector quantization (VQ) typically partitions a full weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ into a set of d_g-dimensional vectors $\mathbf{X} = \{\mathbf{x}_i \in \mathbb{R}^{d_g}\}$ in the continuous space and maps these vectors to one codebook vector from a finite set of candidates $\mathcal{C}$.

$$\begin{aligned}\mathbf{X} = \{\mathbf{x}_i = \mathbf{W}_{i,\, j:j+d_g} | i = 0, \cdots, d_{\text{out}} - 1, \\ j = 0, \cdots, d_{\text{in}} - 1, j \bmod d_g = 0\}.\end{aligned} \tag{1}$$

Here, d_{out} and d_{in} denote the output and input dimensions of the weight matrix $\mathbf{W}$, respectively. $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K\}$ is obtained using K-means clustering on $\mathbf{X}$, where any centroid $\mathbf{c}_k \in \mathbb{R}^{d_g}$ is adopted as a codebook vector (or codeword for brevity). The number of codewords in $\mathcal{C}$ is determined by the presumed bit-width d_B as $K = 2^{d_B}$. Given any vector $\mathbf{x}_i$ in $\mathbf{X}$, the quantization process approximates it with the nearest codeword in $\mathcal{C}$.

$$q(\mathbf{X}) = \{\hat{\mathbf{x}}_i = \mathbf{c}_{k^*} \mid k^* = \arg\min_k \|\mathbf{x}_i - \mathbf{c}_k\|_2^2\}. \tag{2}$$

In general, VQ assigns each vector to its closest codeword, and yields a high compression ratio. However, a single codebook has limited capacity. When K is fixed, quantization error exponentially grows by increasing the vector dimension d_g, due to the well-known *curse of dimensionality* (Bishop & Nasrabadi, 2006). To address this, **residual vector quantization** (RVQ) (Egiazarian et al., 2024; Xu et al., 2025) is commonly adopted to recursively employ the VQ process to the quantization residuals. The approximation is refined stage by stage in RVQ to achieve an exponential reduction in quantization error with a smaller total codebook size than single-level VQ.

3.2 EARTH MOVER'S DISTANCE

Common approaches for measuring the similarity between two codebooks include KL divergence and cosine similarity. However, these metrics do not adequately account for the utilization rate of individual codewords in matching two codebooks. They are usually degraded, when the two distributions differ substantially in shape. The Earth Mover's Distance (EMD) (Rubner et al., 2000) provides a principled way to measure the minimal cost of transforming one distribution into another. EMD is proven particularly effective for the similarity-based codebook merging introduced in this paper.

Formally, given two discrete distributions represented as weighted point sets $P = \{(\mathbf{p}_i, w_i^p)\}_{i=1}^n$ and $Q = \{(\mathbf{q}_j, w_j^q)\}_{j=1}^m$ with $\sum_i w_i^p = \sum_j w_j^q$, the EMD is defined as the minimum cost of

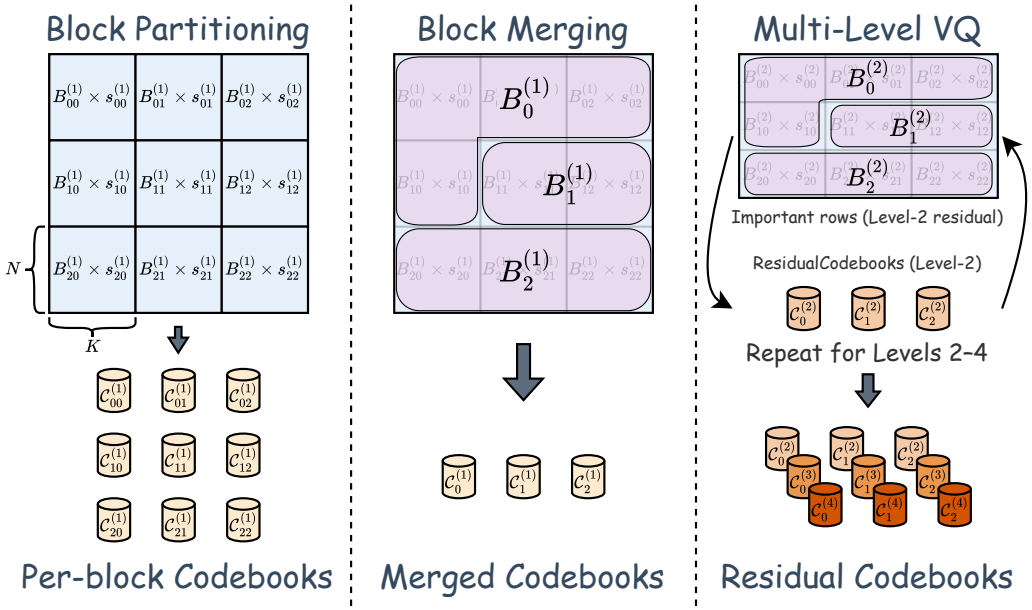

Figure 1: Overview of MBVQ. The left panel illustrates the partitioning of the full weight matrix into fixed-size blocks of size $N \times K$, where each block is assigned an independent scaling factor s. The middle panel shows the block merging strategy based on similarity, which effectively reduces the number of codebooks. The right panel presents the multi-level residual vector quantization process guided by row-wise importance.

transforming P into Q (Pele & Werman, 2009):

$$\mathrm{EMD}(P,Q) = \min_{\mathbf{F}\geq 0} \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}\cdot d(\mathbf{p}_i,\mathbf{q}_j)}{\sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}} \tag{3}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} f_{ij} \leq w_i^p,\ \forall i, \tag{4}$$

$$\sum_{i=1}^{n} f_{ij} \leq w_j^q,\ \forall j, \tag{5}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij} = \min\Big(\sum_i w_i^p, \sum_j w_j^q\Big), \tag{6}$$

where f_{ij} denotes the flow from $\mathbf{p}_i$ to $\mathbf{q}_j$, and $d(\cdot,\cdot)$ is a ground distance such as Euclidean distance.

When both distributions have equal total mass and the ground distance is a metric, the EMD admits a dual formulation that is equivalent to the 1-Wasserstein distance (Arjovsky et al., 2017).

$$\mathrm{EMD}(P,Q) = \sup_{\|\phi\|_{\mathrm{Lip}}\leq 1} \int \phi \,\mathrm{d}P - \int \phi \,\mathrm{d}Q. \tag{7}$$

4 Proposed Method

This section presents the proposed multi-level block-wise vector quantization, as illustrated in Figure 1. We begin with block-wise partitioning of weight matrices into small independent quantization units to alleviate the memory bottleneck in Section 4.1, and then introduce a block merging strategy that allows row-adjacent blocks to share codebooks based on the entropy and Earth Mover's Distance metrics in Section 4.2. In Section 4.3, we develop the multi-level residual quantization

scheme by incorporating second-order row-wise importance scores to recover performance. Finally, Section 4.4 presents a per-block rescaling technique to maximize numerical precision.The overall algorithm is provided in Appendix A.1.

4.1 BLOCK-WISE PARTITIONING

Existing vector quantization methods typically treat the entire weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ as a single quantization unit with a large global codebook (*e.g.*, $2^{15} \times 8$ in Egiazarian et al. (2024)). During dequantization, such a codebook often exceeds on-chip memory capacity, triggering costly off-chip memory accesses Chowdhury (2026).

To address this, we partition $\mathbf{W}$ along both row and column dimensions into blocks of size $N \times K$. Each block serves as an independent quantization unit with its own capacity-constrained codebook. Specifically, each block yields $N \times (K/d_g)$ vectors for codebook initialization, and the per-block codebook contains 2^{d_B} codewords (d_B is the encoding bit-width). A corresponding encoding matrix of shape $N \times (K/d_g)$ stores the d_B-bit code indices. In total, the full matrix produces $(d_{\text{out}}/N) \times (d_{\text{in}}/K)$ codebooks. After initializing all codebooks independently, we concatenate the encoding matrices to preserve the original matrix shape. Constraining each codebook to a small size controls the total codebook overhead and maintains a competitive compression ratio.

During dequantization, we operate at block granularity: only one small codebook needs to be loaded at a time, and the full weight matrix is reconstructed by concatenating the dequantized blocks.

4.2 BLOCK MERGING

After partitioning the weight matrix into blocks, we observe that in most LLaMA layers, the weight distribution exhibits a row-wise similarity pattern, as shown in Figure 2 : values along the output channel dimension share similar distributional characteristics, leading to high similarity between codebooks of row-adjacent blocks. Motivated by this observation, we introduce a block merging strategy that allows row-adjacent blocks to share a common codebook, thereby further improving the compression ratio while reducing dequantization overhead.

We determine whether two adjacent blocks should be merged by jointly considering first-order entropy and the EMD. First, we compute the normalized entropy of every block to detect codebook collapse. Let $\mathcal{C}_k$ be the codebook of block k, and let $p(c)$ denote the utilization probability of codeword $c \in \mathcal{C}_k$ (computed as the fraction of vectors assigned to c). The first-order entropy is defined as $H(\mathcal{C}_k) = -\sum_{c \in \mathcal{C}_k} p(c) \log p(c)$, and the maximum possible entropy given $|\mathcal{C}_k|$ active codewords is $H_{\max}(\mathcal{C}_k) = \log |\{c \in \mathcal{C}_k : p(c) > 0\}|$. The normalized entropy is then:

$$\tilde{H}(\mathcal{C}_k) = \frac{H(\mathcal{C}_k)}{H_{\max}(\mathcal{C}_k)}, \tag{8}$$

where we define $\tilde{H}(\mathcal{C}_k) = 0$ if at most one codeword is used (i.e., the codebook has effectively collapsed). A codebook with $\tilde{H}(\mathcal{C}_k) < \tau$ is identified as collapsed and is required to merge with its most similar adjacent block.

Second, we compute pairwise similarity between all row-adjacent blocks, where the last block of a row is considered adjacent to the first block of the next row (row-major wraparound order). For two codebooks $\mathcal{C}_i$ and $\mathcal{C}_j$ with utilization weight vectors $\mathbf{w}_i$ and $\mathbf{w}_j$, we first compute the exact Wasserstein distance $\mathcal{W}(\mathcal{C}_i, \mathcal{C}_j)$ by solving the optimal transport problem:

$$\mathcal{W}(\mathcal{C}_i, \mathcal{C}_j) = \min_{\mathbf{F} \geq 0} \sum_{a=1}^{|\mathcal{C}_i|} \sum_{b=1}^{|\mathcal{C}_j|} f_{ab} \cdot \|\mathbf{c}_a - \mathbf{c}_b\|_2, \tag{9}$$

subject to the marginal constraints $\sum_b f_{ab} = w_i^a$ and $\sum_a f_{ab} = w_j^b$.

To account for the intrinsic dispersion of each codebook, we normalize the Wasserstein distance by the average intra-codebook distance. For codebook $\mathcal{C}$ with codewords $\{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$, the mean intra-codebook distance is defined as:

$$\bar{d}(\mathcal{C}) = \frac{2}{n(n-1)} \sum_{1 \leq a < b \leq n} \|\mathbf{c}_a - \mathbf{c}_b\|_2. \tag{10}$$

The average intra-distance of the two codebooks is $\bar{d}_{\text{avg}} = (\bar{d}(\mathcal{C}_i) + \bar{d}(\mathcal{C}_j))/2$. The normalized Wasserstein distance is then $\mathcal{W}_{\text{norm}} = \mathcal{W}/\bar{d}_{\text{avg}}$, and the final similarity score is defined as:

$$\text{Sim}(\mathcal{C}_i, \mathcal{C}_j) = \max\big(0,\, 1 - \mathcal{W}_{\text{norm}}\big), \tag{11}$$

with $\text{Sim}(\mathcal{C}_i, \mathcal{C}_j) = 1$ if $\bar{d}_{\text{avg}} < 10^{-8}$ (i.e., both codebooks are nearly degenerate). We then select the top α most similar adjacent block pairs for merging. After merging, codebooks are re-initialized for the resulting merged blocks, producing updated codebooks and encoding matrices.

Since merging operates solely on codebook assignment and does not alter the shape of the encoding matrix, it reduces the number of codebooks while preserving the encoding structure. This translates directly to an improved compression ratio and fewer codebook loads during dequantization.

4.3 Multi-Level Quantization

Following the approach of AQLM (Egiazarian et al., 2024) and CRVQ (Xu et al., 2025), we observe that block-wise quantization alone incurs a significant performance penalty. To address this, we introduce a multi-level residual quantization scheme. At level 1, we quantize the original weight matrix directly. For each subsequent level $\ell \geq 2$, we compute the residual between the original weight matrix and the accumulated quantized reconstruction from all previous levels, and then quantize this residual. Formally, let $\mathbf{W}$ be the original weight matrix and $\hat{\mathbf{W}}^{(\ell)}$ be the quantized reconstruction at level ℓ. The accumulated reconstruction up to level $\ell - 1$ is $\hat{\mathbf{W}}_{\text{acc}}^{(\ell-1)} = \sum_{t=1}^{\ell-1} \hat{\mathbf{W}}^{(t)}$, and the residual at level ℓ is defined as:

$$\mathbf{R}^{(\ell)} = \mathbf{W} - \hat{\mathbf{W}}_{\text{acc}}^{(\ell-1)}, \tag{12}$$

which serves as the target to be quantized at level ℓ.

At each additional level, we compute row-wise importance scores based on the residual error. Let $\Delta^{(\ell-1)} = \mathbf{W} - \hat{\mathbf{W}}_{\text{acc}}^{(\ell-1)}$ be the residual. We compute importance scores using the second-order information captured by the layer-wise input correlation matrix $\mathbf{X}^\top\mathbf{X}$ (Frantar & Alistarh, 2022; Frantar et al., 2022), where $\mathbf{X}$ is the layer input obtained from a calibration dataset. The importance score for the i-th row is defined as:

$$s_i = \frac{1}{d_{\text{in}}} \cdot \boldsymbol{\delta}_i^\top \mathbf{X}^\top \mathbf{X} \boldsymbol{\delta}_i, \tag{13}$$

where $\boldsymbol{\delta}_i \in \mathbb{R}^{d_{\text{in}}}$ is the i-th row of $\Delta^{(\ell-1)}$, and the factor $1/d_{\text{in}}$ computes the mean over the input dimension. We then reshape the scores into groups of N consecutive rows and, within each group, select the rows with the highest s_i for residual quantization. In our experiments, this row-wise, second-order importance estimation consistently outperforms both column-wise alternatives and simple L2-based heuristics. The codebook size at each subsequent level is identical to that of level 1, while the encoding matrix is reduced to cover only the selected rows. Owing to the small fraction of selected rows and modest codebook size, this multi-level design incurs only a limited storage overhead.

4.4 Scale Design

Under residual quantization, higher-level residual weights can become excessively small in magnitude. Moreover, since each block's codebook size is constrained, every initialization step compresses a large number of vectors into a relatively tiny codebook, making it essential to fully utilize the available numerical precision. Inspired by the scaling factors used in scalar quantization (Esser et al., 2019; Lin et al., 2024), we introduce a per-block scale factor that rescales the values within each block so that their absolute magnitudes are centered around 1.

Concretely, we initialize one scale factor per block (equivalently, per codebook) at every level. For each block $\mathbf{B} \in \mathbb{R}^{N \times K}$ at level ℓ, the scale factor s is initialized as:

$$s = \exp\big(-\text{median}\big(\{\log|b_{ij}| : b_{ij} \in \mathbf{B}, b_{ij} \neq 0\}\big)\big), \tag{14}$$

which aligns the median absolute value in the logarithmic domain to 1. Each scale is clipped to the range:

$$s \in [10^{-4},\, 10^{4}], \tag{15}$$

to prevent numerical instability during training.

Model	Method	Wbits	Wiki2↓	C4↓	Wino.↑	Hella.↑	PIQA↑	BoolQ↑	ARC-e↑	ARC-c↑	Avg.↑
LLaMA2-7B	FP16	16.0	5.12	6.63	69.22	75.93	79.11	77.68	74.75	46.16	70.47
	QTIP	1.50	7.69	10.81	65.77	53.68	72.36	73.64	68.91	39.75	62.35
	AQLM	1.52	9.14	10.87	56.08	38.76	63.35	61.41	44.36	23.73	47.95
	GPTVQ	1.75	16.78	36.68	54.22	31.03	59.25	62.11	38.22	20.39	44.20
	MBVQ	1.51	**7.46**	**10.30**	**68.98**	**57.17**	**78.07**	**77.71**	**76.39**	**43.52**	**66.97**
LLaMA3-8B	FP16	16.0	5.75	8.32	72.69	60.19	79.71	81.28	80.13	50.43	70.73
	QTIP	1.50	20.52	24.06	**63.29**	41.82	**69.85**	68.89	63.67	**31.74**	56.54
	AQLM	1.52	16.29	17.34	61.96	42.17	69.42	74.10	63.51	29.78	56.82
	GPTVQ	1.75	34.20	137.51	50.20	27.89	55.50	42.42	34.05	17.32	37.90
	MBVQ	1.51	**11.83**	**17.15**	61.77	**43.65**	69.55	**74.43**	**65.35**	31.72	**57.75**
LLaMA3.2-1B	FP16	16.0	9.07	12.03	61.09	48.14	74.92	63.79	66.20	30.97	57.52
	QTIP	1.50	30.00	34.46	53.19	34.34	65.50	57.64	51.76	24.57	47.83
	AQLM	1.57	26.69	32.63	57.81	41.42	70.89	61.60	61.01	29.41	53.69
	GPTVQ	1.75	353.70	1020.86	48.38	26.38	54.35	45.47	29.08	18.09	36.96
	MBVQ	1.51	**23.78**	**32.53**	**60.62**	**47.72**	**74.43**	**63.98**	**65.49**	**31.31**	**57.26**
LLaMA3.2-3B	FP16	16.0	7.28	10.01	69.69	55.37	76.88	72.81	74.54	42.41	65.28
	QTIP	1.50	16.97	21.42	**59.58**	38.23	**68.20**	**66.79**	57.42	27.94	**53.02**
	AQLM	1.54	23.75	21.78	58.01	37.61	65.56	65.96	56.94	25.94	51.67
	GPTVQ	1.75	44.93	153.70	51.78	28.52	55.82	47.89	34.26	18.52	39.47
	MBVQ	1.51	**15.27**	**21.01**	56.75	**38.25**	67.36	66.30	**58.96**	**28.07**	52.61
Mistral-7B	FP16	16.0	4.91	7.42	73.95	61.24	80.58	83.82	80.72	50.17	71.74
	QTIP	1.50	7.35	11.20	62.56	45.91	68.55	66.16	61.28	32.42	56.15
	AQLM	1.54	8.27	11.50	59.57	38.24	57.43	54.17	37.69	20.73	44.64
	GPTVQ	1.75	29.16	155.82	49.25	27.82	54.68	38.41	33.63	18.69	37.08
	MBVQ	1.51	**7.08**	**10.80**	**64.25**	**47.96**	**73.94**	**71.10**	**70.03**	**35.58**	**60.48**
Qwen2-1.5B	FP16	16.0	8.87	12.20	63.06	50.84	76.39	78.04	76.81	43.43	64.76
	AQLM	1.54	19.47	34.59	**57.06**	33.32	61.86	62.17	50.63	22.44	47.91
	GPTVQ	1.75	–	–	48.93	25.64	54.41	43.36	23.53	23.72	36.60
	MBVQ	1.51	**17.59**	**25.49**	56.04	**34.74**	**65.56**	**62.75**	**54.21**	**23.98**	**49.55**

Table 1: Main results of the evaluation experiments. Bold denotes the **best** result, and "-" indicates that the result is too large to be meaningful.

5 Experimental Results

5.1 Settings

Models and Data We evaluate MBVQ on LLaMA2-7B, LLaMA3.2-1B/3B, LLaMA3-8B, Mistral-7B, and Qwen2-1.5B. The calibration dataset is RedPajama, from which we sample 4096 sequences of text length 4096.

Baselines and Configuration We compare MBVQ against post-training quantization (PTQ) methods designed for extreme low-bit compression, including AQLM (Egiazarian et al., 2024), GPTVQ (van Baalen et al., 2025) and QTIP (Tseng et al., 2024b). Since MBVQ operates at an equivalent bit-width of approximately 1.5 bits per parameter, we configure all baseline methods to a comparable bit-width. To balance compression ratio and model quality, we set the hyperparameters of MBVQ as follows: block size $N = 256$, $K = 1024$; codebook vector dimension $d_g = 4$, encoding bit-width $d_B = 6$; number of quantization levels $m = 4$; channel importance ratio $\lambda = 0.05$; block merging ratio $\alpha = 0.75$; and codebook collapse threshold $\tau = 0.5$.

Evaluation Metrics To evaluate model quality, we measure perplexity on the WikiText2 (Merity et al., 2017) and C4 (Raffel et al., 2020) datasets, using evaluation sequences of the same length as those used for calibration. In addition, we evaluate zero-shot accuracy on a suite of commonsense reasoning benchmarks, including WinoGrande (Sakaguchi et al., 2021), HellaSwag Zellers et al.

Block Config $(N \times K)$	256×4096	1024×1024	4096×256	32×4096	256×1024-m	256×4096-w/o s
Perplexity (PPL)	5.1490	5.1529	5.1556	5.1496	5.1464	5.1622
ΔPPL (w.r.t. FP16)↑	0.0325	0.0364	0.0391	0.0331	0.0299	0.0457

Table 2: Experimental results on block configuration. The suffix 'm' denotes the use of the block merging strategy, and 'w/o s' indicates that the scaling strategy is not applied.

Strategy	**Mean**	**Weighted**	**Self**	**Max**	**None**
PPL	5.1464	5.1511	5.1518	5.1479	5.1490
ΔPPL↑	0.0299	0.0346	0.0353	0.0314	0.0325

Table 3: Perplexity results when using different EMD normalization methods.

(2019), PIQA Bisk et al. (2020), BoolQ (Clark et al., 2019), ARC-e/ARC-c Clark et al. (2018). All evaluations are conducted using the open-source toolkit LM-Evaluation-Harness[1].

5.2 Main Results

As shown in Table 1, MBVQ consistently outperforms the baselines across models of varying scales under comparable compression ratios. Notably, on larger models such as LLaMA2-7B, our approach achieves strong performance with an average perplexity increase of only 3.00, indicating that it is well-suited for compressing larger-scale models. In terms of zero-shot accuracy, MBVQ attains the best results across all benchmarks and substantially surpasses the competing approaches.

6 Discussion

This section presents additional experiments and analyses of our proposed method. Through extensive experiments, we observe that quantizing different layers of LLaMA2-7B has varying effects on the final model quality. We find that Layer 1 exhibits substantially higher quantization sensitivity than the remaining layers. Accordingly, all experiments in this section are conducted with Layer 1 of LLaMA2-7B as the quantization baseline.

6.1 Block Configuration

Under a joint consideration of compression ratio and inference speedup, we set the total block size to $N \times K = 2^{20}$. On this basis, we analyze the effect of stretching blocks along the row versus the column dimension. The corresponding results are shown in Table 2. Empirically, our chosen configuration of $K > N$ outperforms $K < N$ by an average of 20.18% and surpasses the square block setting $N = K$ by 11.89%. As shown in Figure 2, both the original model weights and the residual weights exhibit stronger distributional similarity along the output channel dimension. This finding aligns with the row-wise similarity observation discussed in Section 4.2 and supports our adoption of horizontally elongated block shapes. We further test two extreme settings, $N \times K = 256 \times 4096$ and $N \times K = 32 \times 4096$, under per-block scaling. The latter outperforms the former by only 1.8%, suggesting that under per-block scaling, the average reconstruction error across blocks is sufficiently narrowed such that further reducing block size yields diminishing returns.

6.2 Block Merging Strategy

When computing the normalized EMD distance, we evaluate four normalization variants: normalization by the maximum intra-codebook distance, the mean intra-codebook distance, the weighted mean intra-codebook distance, and self-distance. The corresponding results are shown in Table 3. Among

[1] https://github.com/EleutherAI/lm-evaluation-harness

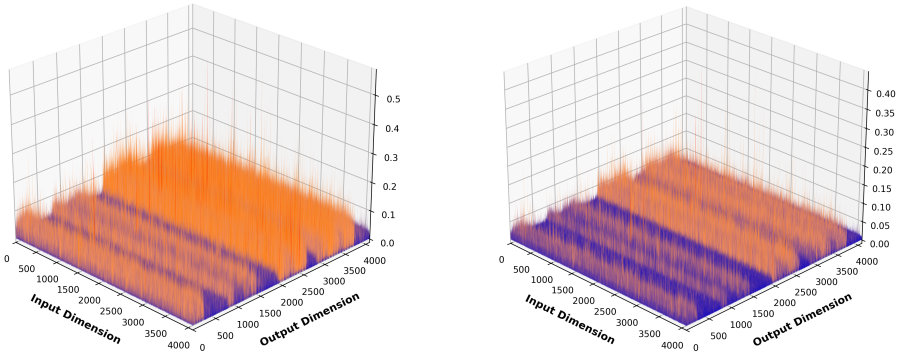

Figure 2: Weight distribution of the `self_attn.k_proj` layer in Layer 1 of LLaMA2-7B (left) and the quantization residual distribution at level 2 (right).

these, normalization by the mean intra-codebook distance achieves the best performance, outperforming weighted mean normalization by 17.80%, self-distance normalization by 20.45%, and maximum distance normalization by 5.68%. To further validate the effectiveness of our block merging strategy, we compare two configurations at the same effective block size: $N \times K = 256 \times 4096$ without merging and $N \times K = 256 \times 1024$ with 75% of blocks merged. Under identical compression ratios and improved inference speedup, the merging-enabled configuration surpasses the non-merging baseline by 8.68% in model quality.

6.3 Hardware-Friendly Design and Speedup Analysis.

Conventional vector quantization methods typically employ codebooks that are too large to reside in on-chip memory, incurring substantial dequantization overhead at inference time. MBVQ addresses this critical limitation by partitioning the weight matrix into blocks and explicitly constraining the per-block codebook size. By flexibly tuning key parameters such as the block dimensions $N \times K$ and the number of quantization levels m, our approach can adapt to diverse model scales and deployment hardware. Under the same equivalent bit-width, MBVQ achieves approximately $6.62\times$ inference speedup while maintaining model accuracy. The speedup is defined as

$$\text{Speedup} = \frac{T_{\text{load_original}}}{T_{\text{load_codebook}} + T_{\text{lookup}}}, \tag{16}$$

where $T_{\text{load_original}} = (N \times K \times 16 \text{ bits}) \,/\, \text{memory bandwidth}$ is the time to load the full FP16 weight block matrix, $T_{\text{load_codebook}} = (m \times d_g \times d_B \times 16 \text{ bits}) \,/\, \text{memory bandwidth}$ is the time to load the small per-block codebooks whose capacity is explicitly constrained, and $T_{\text{lookup}} = \text{MAX}\{(m \times N \times (K/d_g) \times d_B) \,/\, \text{memory bandwidth}, m \times (K/d_g) \times T_{\text{per_lookup}}\}$ is the table lookup time, where $T_{\text{per_lookup}}$ denotes the per-element lookup time, and the lookup operation can be fully parallelized on modern GPUs and NPUs. Because each codebook is small enough to reside in on-chip memory, the dominant memory-access cost is drastically reduced. This analytical speedup is architecture-agnostic and depends primarily on memory bandwidth and lookup parallelism, both of which are widely supported across GPU families. While some edge devices may have limited lookup support, we expect meaningful acceleration across most modern hardware platforms.

7 Conclusion

We presented MBVQ, a hardware-friendly multi-level block-wise vector quantization method for compressing large language models. By partitioning weight matrices into blocks with capacity-constrained codebooks and merging redundant adjacent blocks, MBVQ substantially reduces memory access overhead during inference. Multi-level residual quantization with per-block scaling further recovers model accuracy. Experiments across LLMs demonstrate that MBVQ consistently outperforms existing methods under comparable compression ratios, achieving both better model quality and significant inference speedup.

ETHICAL STATEMENT

Our work focuses on efficient model compression and does not involve human subjects, crowdsourced annotation, or personally identifiable information. All experiments use publicly available pretrained models (LLaMA2, LLaMA3, LLaMA3.2, Mistral, Qwen) and open-source datasets (RedPajama, WikiText2, C4, and standard commonsense reasoning benchmarks), used in accordance with their respective licenses. While our method facilitates broader deployment of LLMs on resource-constrained devices with improved inference efficiency, we acknowledge that any technology lowering deployment barriers could potentially be misused. We encourage responsible deployment and usage monitoring. No AI writing assistance was used beyond language polishing.

AI USE STATEMENT

In this work, we used generative AI tools for translation and language polishing of the manuscript. We have not used generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, cleaning or reformatting datasets, supporting qualitative or thematic data analysis, or interpreting results. The remaining required-disclosure tasks are not applicable to this work. Additionally, we did not use generative AI tools for the recommended-disclosure tasks, including creating or modifying scientific figures or images, suggesting experimental parameters, creating or editing software code, drafting parts of the research paper, summarizing or analyzing existing literature, discovering research topics or identifying gaps, brainstorming, sourcing or searching for information, editing the paper to improve readability, identifying relevant literature, formatting references, suggesting a structure for the paper, or proposing a title or keywords. We have reviewed all AI-assisted work. Specifically, the AI-assisted translation and polishing were verified by the authors against the original content to ensure that no technical claims, numerical results, or citations were altered. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. Pmlr, 2017.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in neural information processing systems*, 36:4396–4429, 2023.

Yi-Dong Chen, Kai-Jun Zheng, Zhen-Hua Guo, Qi-Hao Zhang, Yong-Hua Zhang, and Ji-Dong Zhai. A survey of quantization in llm: Unlocking potential hardware efficiency. *Journal of Computer Science and Technology*, 41(1):341–358, 2026.

Rajarshi Chowdhury. I/o for llm inference: A survey of storage and memory bottlenecks. 2026.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pp. 2924–2936, 2019.

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

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*, 2024.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*, pp. 291–326. Chapman and Hall/CRC, 2022.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hongyaoxing Gu, Lijuan Hu, Liye Yu, Haowei Li, and Fangfang Liu. Lopro: Enhancing low-rank quantization via permuted block-wise rotation. *arXiv preprint arXiv:2601.19675*, 2026.

Chun Hu, Junhui He, Shangyu Wu, Yuxin He, Chun Jason Xue, and Qingan Li. Mlwq: Efficient small language model deployment via multi-level weight quantization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8078–8088, 2025.

Dongwon Jo, Taesu Kim, Yulhwa Kim, and Jae-Joon Kim. Mixture of scales: Memory-efficient token-adaptive binarization for large language models. *Advances in Neural Information Processing Systems*, 37:137474–137494, 2024.

Gilhyeon Lee, Seonggeun Kim, Dongjun Lee, Kyungmin Goh, and Hyun Kim. Towards efficient language giants: A comprehensive survey on structural optimizations and compression techniques for large language models. *Neural Networks*, 201:108900, 2026. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2026.108900. URL https://www.sciencedirect.com/science/article/pii/S0893608026003618.

Sangjun Lee, Seung-taek Woo, Jun-gyu Jin, Changhun Lee, and Eunhyeok Park. 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*, pp. 35520–35538, 2025.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100, 2024.

Fangxin Liu, Zongwu Wang, Jinhong Xia, Junping Zhao, Shouren Zhao, Jinjin Li, Jian Liu, Li Jiang, and Haibing Guan. FlexQuant: A flexible and efficient dynamic precision switching framework for LLM quantization. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 4152–4161, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.221. URL https://aclanthology.org/2025.findings-emnlp.221/.

Yifei Liu, Jicheng Wen, Yang Wang, Shengyu Ye, Li Lyna Zhang, Ting Cao, Cheng Li, and Mao Yang. VPTQ: Extreme low-bit vector post-training quantization for large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8181–8196, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.467. URL https://aclanthology.org/2024.emnlp-main.467/.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 467–484, 2024b.

Zihan Liu, Xinhao Luo, Junxian Guo, Wentao Ni, Yangjie Zhou, Yue Guan, Cong Guo, Weihao Cui, Yu Feng, Minyi Guo, Yuhao Zhu, Minjia Zhang, Chen Jin, and Jingwen Leng. Vq-llm: High-performance code generation for vector quantization augmented llm inference. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1496–1509, 2025b. doi: 10.1109/HPCA61900.2025.00112.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *2009 IEEE 12th international conference on computer vision*, pp. 460–467. IEEE, 2009.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *Proceedings of machine learning research*, 235:48630, 2024a.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De. Qtip: Quantization with trellises and incoherence processing. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 59597–59620. Curran Associates, Inc., 2024b. doi: 10.52202/079017-1904. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/6de2e84b8da47bb2eb5e2ac96c63d2b0-Paper-Conference.pdf.

Mart van Baalen, Andrey Kuzmin, Ivan Koryakovskiy, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. Gptvq: The blessing of dimensionality for llm quantization, 2025. URL https://arxiv.org/abs/2402.15319.

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

Yuzhuang Xu, Shiyu Ji, Qingfu Zhu, and Wanxiang Che. Crvq: Channel-relaxed vector quantization for extreme compression of llms. *Transactions of the Association for Computational Linguistics*, 13:1488–1506, 2025.

Zukang Xu, Xing Hu, Qiang Wu, and Dawei Yang. Rsavq: Riemannian sensitivity-aware vector quantization for large language models. *Advances in Neural Information Processing Systems*, 38: 1409–1436, 2026.

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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yuxuan Yue, Zukang Xu, Zhihang Yuan, Dawei Yang, Jianlong Wu, and Liqiang Nie. Pcdvq: Enhancing vector quantization for large language models via polar coordinate decoupling. *arXiv preprint arXiv:2506.05432*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Binrui Zeng, Yongtao Tang, Bin Ji, Xiaodong Liu, Xiaopeng Li, and Jie Yu. Swsc: Compressing large language models via sharing weights for similar channels. In *Proceedings of the 2025 9th International Conference on Computer Science and Artificial Intelligence*, pp. 132–139, 2025.

Xi Zhang, Xiaolin Wu, Jiamang Wang, and Weisi Lin. Learning grouped lattice vector quantizers for low-bit llm compression. *Advances in Neural Information Processing Systems*, 38:110672–110698, 2026.

A Appendix

A.1 Overall Algorithm

The proposed MBVQ algorithm is outlined in Algorithm 1. For each linear layer weight matrix $\mathbf{W}$ in every Transformer block, we perform the following steps. First, we partition $\mathbf{W}$ into blocks according to fixed block size parameters N and K, and then apply the block merging procedure described in Sections 4.2 to all blocks (Lines 3–4). Next, we quantize the weight matrix level by level. At level 1, the input is the original weight matrix; at subsequent levels, the input is the quantization residual from the preceding level. At each level, we compute row-wise importance scores and select the top λ fraction of rows for selective residual vector quantization. Prior to quantization at each level, per-block scaling factors are computed and used to rescale the effective quantization weights for that level. After completing quantization at a level, we fine-tune the corresponding codebook and perform beam search optimization to obtain improved codebooks and encoding assignments (Lines 5–14). Once codebook optimization is complete across all levels, we jointly fine-tune all codebooks and scales, and perform beam search over all encoding assignments (Lines 15–20).

Following AQLM (Egiazarian et al., 2024), we subsequently fine-tune each Transformer block. Furthermore, after the entire model has been quantized, we perform end-to-end fine-tuning of the full quantized model (Tseng et al., 2024a; Egiazarian et al., 2024) (Lines 22 and 24).

Algorithm 1 MBVQ: Multi-Level Block-Wise Vector Quantization

Require: A pretrained LLM $\mathcal{M}$ with Transformer blocks $\{T_1, T_2, \ldots, T_k\}$, block size $N \times K$, number of quantization levels m, channel importance ratio λ, block merging ratio α, codebook collapse threshold τ, quantization loss threshold ϵ, layer inputs $\mathbf{X}$ and $\mathbf{X}^\top\mathbf{X}$ from a calibration dataset.

for `Transformer block` $T_i \in \{T_1, T_2, \ldots, T_k\}$ **do**
  for `weight` $\mathbf{W} \in$ `LinearLayerWeights`(T_i) **do**
    $\mathbf{W}_b, \mathbf{C}_b, \mathbf{I}_b \leftarrow$ `InitBlock`$(\mathbf{W}, N, K)$
    $\mathcal{B} \leftarrow$ `MergeBlock`$(\mathbf{W}_b, \mathbf{C}_b, \mathbf{I}_b, \tau, \alpha)$
    $\{\mathcal{B}$: non-uniform block shapes after merging$\}$
    for $l = 1, \ldots, m$ **do**
      $\mathbf{Imp} \leftarrow$ `RowImportance`$(\mathbf{W}, \mathbf{W}_q^{(l-1)}, \mathbf{X}^\top\mathbf{X})$
      $\mathbf{W}^{(l)} \leftarrow$ `GetResidual`$(\mathbf{W}, \mathbf{W}_q^{(l-1)}, \mathbf{Imp}, \lambda)$
      $\mathbf{W}_s^{(l)}, \mathbf{S}_l \leftarrow$ `GetScales`$(\mathbf{W}^{(l)}, N, K)$
      $\mathbf{W}_q^{(l)}, \mathbf{C}_l, \mathbf{I}_l \leftarrow$ `BlockVectorQuant`$(\mathbf{W}_s^{(l)}, \mathcal{B})$
      while `QuantLoss`$(\mathbf{W}, \mathbf{W}_q^{(l)}, \mathbf{X}^\top\mathbf{X}) \geq \epsilon$ **do**
        `FineTuneCodebook`$(\mathbf{C}_l)$
        `BeamSearchOptimize`$(\mathbf{W}^{(l)}, \mathbf{S}_l, \mathbf{C}_l, \mathbf{I}_l)$
      end while
    end for
    while `QuantLoss`$(\mathbf{W}, \mathbf{W}_q, \mathbf{X}^\top\mathbf{X}) \geq \epsilon$ **do**
      `FineTuneCodebook`$(\{\mathbf{C}_1, \ldots, \mathbf{C}_m\},$
        $\{\mathbf{S}_1, \ldots, \mathbf{S}_m\})$
      `BeamSearchOptimize`$(\mathbf{W}, \{\mathbf{S}_1, \ldots, \mathbf{S}_m\},$
        $\{\mathbf{C}_1, \ldots, \mathbf{C}_m\}, \{\mathbf{I}_1, \ldots, \mathbf{I}_m\})$
    end while
  end for
  `FineTuneTransformer`(T_i)
end for
$\mathcal{M}_q \leftarrow$ `EndToEndFineTune`$(\mathcal{M})$

Ensure: The quantized model $\mathcal{M}_q$.