TriVQ: Joint Vector Quantization of Weights, Activations, and CodeBooks for Large Language Models

Anonymous ACL submission

Abstract

To address the memory bandwidth and computational bottlenecks of Large Language Models (LLMs), we propose TriVQ, a framework for the joint quantization of weights, activations, and codebooks (WAC). While existing low-bit vector quantization methods rely on floating-point matrix multiplications, TriVQ enables pure integer inference by overcoming coarse-grained codebook bottlenecks and compounding multi-source errors. Specifically, TriVQ introduces a Dual-Weighted Loss to optimize codebooks based on usage frequency and activation sensitivity, alongside a Cross-Distribution Calibration mechanism to mitigate coupled errors across transformer layers. Extensive experiments on LLaMA models demonstrate that TriVQ maintains robust language generation and zero-shot reasoning under extreme W2A4C4 configurations—where conventional baselines collapse—while achieving a 3.8x hardware-friendly inference speedup over optimized FP16 baselines.

1 Introduction

Large language models (LLMs) achieve remarkable performance but face high storage and inference overheads due to increasing sizes (Zhai et al., 2022). Often exceeding single-GPU memory, inference is bottlenecked by memory bandwidth (Sheng et al., 2023). Low-bit quantization addresses this, enabling efficient deployment through reduced memory usage and low-precision computation.

Current research on weight-activation quantization primarily focuses on scalar quantization (SQ). Recent methods (Ashkboos et al., 2024; Liu et al., 2025a; Xiao et al., 2023; Shao et al., 2024; Lin et al., 2024) have demonstrated competitive model performance under W4A4 (4-bit weights and 4-bit activations) settings. However, the limited representational capacity of SQ makes it difficult to maintain accuracy under ultra-low-bit weight quantization (e.g., W2). While quantization-aware training (QAT) methods, such as BitNet (Wang et al., 2023), partially alleviate this issue through low-bit pretraining, they typically require training from scratch and incur prohibitive computational cost.

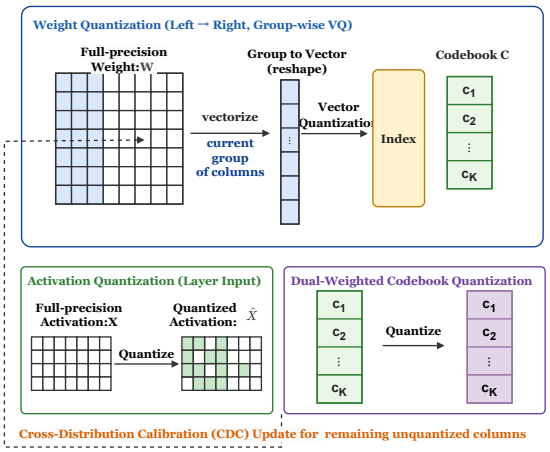

Figure 1: The overall framework of TriVQ, which jointly quantizes weights, activations, and codebooks to enable efficient end-to-end integer inference.

To improve the representation capability of ultra-low-bit weights, recent studies have explored vector quantization (VQ) as an alternative compression paradigm (Egiazarian et al., 2024), achieving promising model performance even at 2-bit precision.

However, existing VQ-based methods are mainly weight-only: their compressed weights are typically dequantized to high-precision formats, such as FP16, before floating-point GEMM operations (Liu et al., 2025b; Tseng et al., 2024). Consequently, they reduce storage and memory traffic but provide limited reduction in arithmetic complexity (Zhao et al., 2024).

This motivates a VQ-based weight-activation quantization framework that combines vector-quantized weights with low-bit activations. By

enabling matrix multiplications to be executed directly in the low-precision integer domain, such a design can reduce both memory access overhead and computational cost, making it more suitable for efficient deployment and scalable serving.

However, deeply integrating VQ with activation quantization to build an efficient end-to-end integer inference framework remains highly challenging. First, the codebooks must also be quantized into low-bit integer representations. Yet, due to the discrete and highly unstructured nature of VQ codebooks, they are difficult to accommodate within hardware-friendly fine-grained quantization schemes, requiring coarse-grained per-tensor mappings, which introduce a structural trade-off between quantization accuracy and execution efficiency. Furthermore, jointly quantizing weights, activations, and codebooks introduces multiple sources of quantization error, including activation truncation under long-tailed distributions, codebook approximation error, and weight quantization loss. Under ultra-low-bit settings, such errors become strongly coupled and progressively accumulate across deep transformer layers and autoregressive decoding steps, leading to significant performance degradation.

To address these challenges, we introduce a unified framework, as illustrated in Fig. 1, and summarize our main contributions as follows:

1. We propose a dual-weighted loss (DWL) for codebook quantization to reduce large errors from coarse-grained per-tensor codebook quantization.

2. We introduce Cross-Distribution Calibration (CDC) for joint weight-activation-codebook (WAC) quantization to mitigate coupled errors across deep transformer layers and autoregressive decoding under extreme low-bit settings.

3. We develop a hardware-friendly joint WAC quantization framework that enables efficient low-precision integer-domain matrix multiplication, overcoming the acceleration limitations of existing VQ methods. The code is available at `https://anonymous.4open.science/r/TriVQ_911/`.

2 Background and Motivation

2.1 Post-Training Quantization in LLM

Post-training quantization (PTQ) has become a key technique for efficient deployment of large language models (LLMs) (Dettmers et al., 2022; Yao et al., 2022; Frantar et al., 2022). By converting floating-point weights $W \in \mathbb{R}^{oc \times ic}$ into low-bit representations, PTQ significantly reduces storage and computation overhead, but inevitably introduces truncation and rounding errors that degrade model performance. To recover model accuracy without expensive full-model fine-tuning, modern PTQ methods typically employ a calibration procedure based on a small calibration dataset. The core objective of this calibration process is to minimize the reconstruction error between the layer outputs before and after quantization, which can be formulated as:

$$\min_{\Delta w} \left\| X(w + \Delta w)^\top - X w^\top \right\|_F^2 \quad \text{s.t. } \Delta w = \hat{w} - w \tag{1}$$

where X denotes the input activations, w and $\hat{w}$ are the full-precision and quantized weights of a single row, respectively, and Δw is the quantization-induced perturbation.

To efficiently solve this objective, advanced PTQ (Frantar et al., 2022) methods leverage second-order Hessian information, following the idea of Optimal Brain Surgeon (OBS) (Hassibi et al., 1993). Given $H^{-1} = (X^\top X)^{-1}$, they select and quantize a target weight dimension q, then compensate the remaining unquantized weights according to:

$$\Delta w = \frac{\hat{w}_q - w_q}{[H^{-1}]_{qq}} \cdot [H^{-1}]_{q,:}. \tag{2}$$

Through iterative quantization and error compensation, quantization errors can be progressively redistributed into the remaining continuous weight space, enabling LLMs to maintain strong performance under low-bit settings.

Unlike weights, activations are harder to quantize due to outliers. Existing methods (Xiao et al., 2023; Liu et al., 2025a) mitigate this issue through smoothing or rotation, enabling competitive weight-activation quantization performance. GPTAQ (Li et al., 2025) further extends weight-only calibration by jointly optimizing weight and activation reconstruction losses. Nevertheless, extending such compensation strategies to vector quantization, while jointly handling the coupled errors from weights, activations, and codebooks, remains a critical challenge addressed in this work.

2.2 Vector Quantization and Limitations of Existing Methods

Vector quantization improves ultra-low-bit representation capability by clustering weight sub-

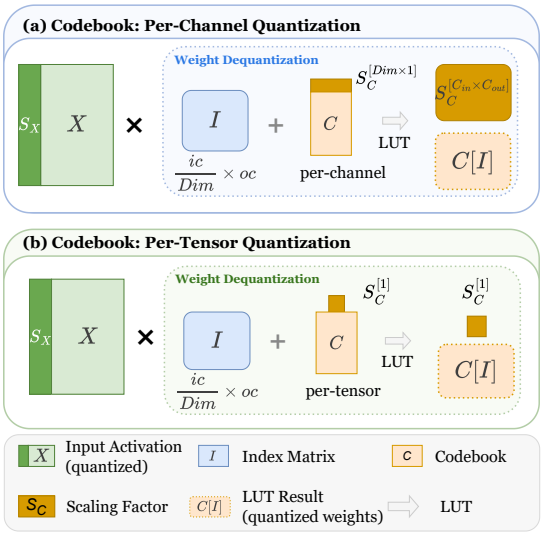

Figure 2: Illustration of per-tensor vs. per-channel quantization for VQ codebooks.

vectors into a shared codebook with low-bit indices (Liu et al., 2024). Specifically, VQ partitions the weight matrix $W \in \mathbb{R}^{oc\times ic}$ into $N = oc \times \frac{ic}{d}$ d-dimensional subvectors $\{w_i\}_{i=1}^{N}$. For simplicity, all subvectors are clustered into a single shared codebook $C \in \mathbb{R}^{K\times d}$, where each codeword captures a representative local weight pattern. Each subvector w_i is then replaced by the index I_i of its nearest codeword, and the approximate weights are reconstructed during inference via Look-Up Table (LUT) operations (Liu et al., 2024).

Benefiting from its ability to capture local structural features, VQ shows clear advantages over scalar quantization at extremely low bit-widths. However, despite its effectiveness in compressing LLM weights and reducing memory bandwidth overhead, integrating VQ with activation quantization for end-to-end integer-only inference still faces two fundamental challenges:

Challenge 1: The Bottleneck of Mandatory Per-Tensor Quantization

To enable a complete Integer GEMM pipeline, both activations X and the codebook C must be quantized into integers ($\mathrm{X_{int}}$ and $\mathrm{C_{int}}$). Ideally, fine-grained strategies like per-channel quantization allocate independent scaling factors across features, accurately characterizing dimension-specific dynamic ranges to minimize quantization errors.

However, applying such fine-grained strategies directly to the VQ framework is problematic. As depicted in Fig. 2, VQ reconstructs weights by assembling discrete codewords from a contiguously stored codebook through LUT operations. Applying per-channel quantization to C implies that each reconstructed weight element $\hat{W}_{ij}$ could require a distinct scaling factor. This imposes a severe storage burden for scaling parameters and significantly increases post-lookup memory and computational overheads, undermining VQ's benefits in compression and bandwidth reduction.

Therefore, per-tensor quantization is applied to the entire codebook, with all codewords sharing a single scaling factor S_C. In this configuration, the matrix multiplication can be expressed as:

$$Y = (\mathrm{X_{int}C_{int}[I]}) \cdot S_Y, \text{ where } S_Y = S_X \cdot S_C \quad (3)$$

As formulated in Eq. (3), the per-tensor strategy simplifies hardware implementation by using scalar scales S_X and S_C. These scales can be factored out of the inner loop and pre-computed as a single output scale S_Y. Thus, the computation core only performs integer matrix multiplication, i.e., $\mathrm{X_{int}C_{int}[I]}$, followed by lightweight dequantization with S_Y. This avoids costly element-wise scaling and is well suited for efficient memory access and integer-only execution.

However, per-tensor quantization is inherently a coarse-grained quantization strategy. Using a single scaling factor makes it difficult to accurately adapt to the substantial dynamic range variations across different codebook dimensions, thereby introducing additional quantization error. This issue becomes particularly severe under ultra-low-bit settings, leading to an intrinsic trade-off between quantization accuracy and hardware efficiency, and ultimately becoming one of the major bottlenecks for achieving both accurate and efficient integer-only inference in VQ systems.

Challenge 2: Compounding and Amplification of Coupled Quantization Errors

Joint ultra-low-bit quantization over WAC constitutes a highly non-convex optimization problem. In practical LLM inference, activation distributions usually exhibit strong long-tail characteristics and severe channel-wise outliers. Low-bit activation quantization therefore introduces non-negligible truncation errors. Meanwhile, integer quantization of codebooks further incurs approximation errors, while VQ itself inevitably introduces representation loss. In a joint quantization framework, these error sources couple and compound.

Without effective calibration, these errors accumulate across network layers and autoregressive decoding steps, severely degrading generation quality and stability at extremely low bit-widths.

3 Proposed Method

3.1 Dual-Weighted Codebook Quantization

In VQ, codebook quantization can be regarded as secondary quantization of the model weights. Since the codebook is a compact representation obtained from clustering, it is highly sensitive to numerical perturbations. Given the clipping bounds γ and β, conventional affine quantization is formulated as:

$$\begin{aligned} \mathrm{C}_{\text{int}} &= \text{clamp}\left(\left\lfloor \frac{\mathbf{C}}{s} + z \right\rceil, 0, 2^b - 1\right) \\ s &= \frac{\beta - \gamma}{2^b - 1}, \; z = -\left\lfloor \frac{\gamma}{s} \right\rceil \\ \hat{\mathbf{C}} &= s(\mathrm{C}_{\text{int}} - z). \end{aligned} \tag{4}$$

where s and z denote the scale and zero point.

However, since the codebook is constrained to per-tensor quantization, conventional MinMax-based range estimation can cause significant performance degradation at 4-bit precision. To identify better clipping boundaries, a common strategy is to search for γ and β by minimizing the codebook reconstruction error:

$$\min_{\gamma,\beta} \left\| \hat{\mathbf{C}} - \mathbf{C} \right\|_F^2. \tag{5}$$

Nevertheless, this objective solely measures the numerical discrepancy between the original and quantized codebooks, entirely ignoring how the quantization error propagates through weight reconstruction and input activations. This inherent flaw leads to a critical objective misalignment: minimizing the local codebook MSE does not necessarily guarantee the optimal end-to-end perplexity. We empirically corroborate this phenomenon in Fig. 3. As illustrated, when searching the clipping threshold of the codebook over $[0.5, 1]$, the threshold yielding the lowest MSE fails to achieve the best WikiText-2 (Merity et al., 2016) PPL.

To address this issue, we reformulate the objective from uniformly weighted codebook error minimization to dual-weighted codebook quantization loss. Specifically, codebook perturbations impact the final output unequally, as determined jointly by their usage frequency during weight reconstruction and the sensitivity of input activations. Consequently, this dual-weighting explicitly drives the codebook quantization process to precisely compensate for the most critical codebook vectors.

Let the weight matrix be approximated by a codebook $\mathbf{C} \in \mathbb{R}^{K \times d}$ and a frozen one-hot assignment

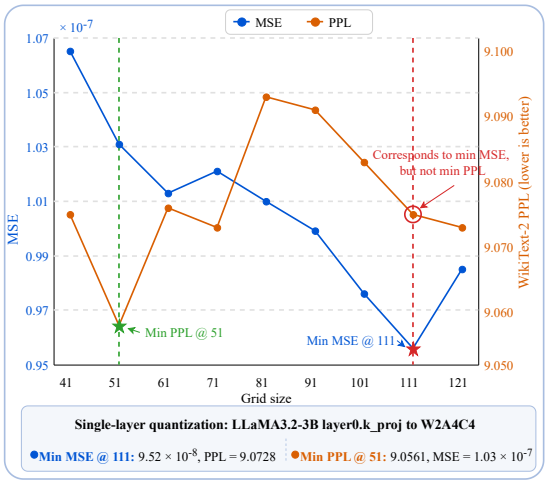

Figure 3: Empirical evidence of objective misalignment in codebook quantization, where local reconstruction error fails to consistently predict end-to-end performance.

matrix $\mathbf{Z} \in \{0,1\}^{K \times N}$, i.e., $\mathbf{W} \approx \mathbf{Z}^\top \mathbf{C}$. Then the layer output can be written as

$$\mathbf{Y} = \mathbf{X}\mathbf{W}^\top \approx \mathbf{X}\mathbf{C}^\top \mathbf{Z}. \tag{6}$$

During codebook quantization, $\mathbf{Z}$ is fixed and the error only comes from the quantized codebook:

$$\Delta\mathbf{C} = \hat{\mathbf{C}}(\theta) - \mathbf{C}, \tag{7}$$

where θ denotes the quantization parameters. The induced output error is

$$\Delta\mathbf{Y} = \mathbf{X}\Delta\mathbf{C}^\top \mathbf{Z}. \tag{8}$$

By vectorization, we obtain

$$\text{vec}(\Delta\mathbf{Y}) = (\mathbf{Z}^\top \otimes \mathbf{X})\text{vec}(\Delta\mathbf{C}^\top). \tag{9}$$

We define $\mathcal{A} = \mathbf{Z}^\top \otimes \mathbf{X}$ as the codebook level error propagation operator. The codebook quantization objective, termed the dual-weighted loss (DWL), is therefore formulated as

$$\min_\theta \mathcal{L}_{\text{DWL}} = \left\| \mathcal{A}\text{vec}(\Delta\mathbf{C}^\top) \right\|_2^2. \tag{10}$$

Equivalently,

$$\mathcal{L}_{\text{DWL}} = \text{vec}(\Delta\mathbf{C}^\top)^\top \mathbf{H}_C \text{vec}(\Delta\mathbf{C}^\top), \tag{11}$$

where

$$\mathbf{H}_C = \mathcal{A}^\top \mathcal{A} = (\mathbf{Z}\mathbf{Z}^\top) \otimes (\mathbf{X}^\top \mathbf{X}) = \mathbf{F} \otimes \mathbf{H}_X. \tag{12}$$

Here, $\mathbf{F} = \mathbf{Z}\mathbf{Z}^\top$ encodes the usage frequency and co-occurrence structure of codebook vectors, while $\mathbf{H}_X = \mathbf{X}^\top \mathbf{X}$ captures input activation sensitivity.

Besides clipping bounds, round-to-nearest is another major source of quantization error, as nearest rounding is not always optimal (Nagel et al., 2020). Since the codebook is much smaller than the original weight matrix, this allows us to introduce a fine-grained adaptive soft-rounding mechanism for codebook quantization. Specifically, we decompose hard rounding into floor rounding and a learnable continuous offset. Given a learnable parameter $\mathbf{V}$, we define a soft-rounding function $h(\mathbf{V}) \in [0, 1]$ to determine whether each codebook element is biased toward rounding down or rounding up. The relaxed integer representation of the codebook is then given by

$$\mathrm{C}_{\mathrm{int}}(\theta) = \mathrm{clamp}\left(\left\lfloor \frac{\mathbf{C}}{s} + z \right\rfloor + h(\mathbf{V}), 0, 2^b - 1\right), \tag{13}$$

where $\lfloor \cdot \rfloor$ denotes the floor operation, and $\theta = \{\gamma, \beta, \mathbf{V}\}$ denotes the continuous parameters optimized during codebook quantization. Under the weighted quantization loss in Eq. (11), the clipping bounds and rounding offsets are jointly optimized, enabling the quantizer to adapt both the quantization range and the rounding directions to better preserve the end-to-end output.

3.2 Cross-Distribution Calibration (CDC) for Joint W-A-C Quantization

3.2.1 Objective Formulation

As discussed above, existing VQ methods mainly target weight-only quantization and rely on an identically distributed calibration assumption. Their optimization objective, $\min \left\|\mathbf{X}\hat{\mathbf{W}}^\top - \mathbf{X}\mathbf{W}^\top\right\|_F^2$, minimizes the current-layer output error by assuming that the input activations remain unchanged after quantization. However, in joint W-A-C quantization, multi-source quantization errors from preceding layers jointly shift the actual layer input from $\mathbf{X}$ to $\hat{\mathbf{X}}$, making such identically distributed calibration ineffective.

To address this issue, we propose cross-distribution calibration (CDC). Instead of assuming identical input distributions, CDC leverages the full-precision output of the current layer($\mathbf{X}\mathbf{W}^\top$) as the reference output. It then optimizes the quantized weights $\hat{\mathbf{W}}$ to align with this ideal output under the shifted activations $\hat{\mathbf{X}}$:

$$\begin{aligned} \min_{\hat{\mathbf{W}}} \ \mathcal{L}_{\mathrm{CDC}} &= \left\|\hat{\mathbf{X}}\hat{\mathbf{W}}^\top - \mathbf{X}\mathbf{W}^\top\right\|_F^2 \\ \text{s.t. } \hat{\mathbf{W}} &\in \Omega(\hat{\mathbf{C}}). \end{aligned} \tag{14}$$

Here, $\mathbf{X}$ denotes the original full-precision activations, $\hat{\mathbf{X}}$ represents the actual shifted activations containing the accumulated quantization errors from preceding layers, and $\Omega(\hat{\mathbf{C}})$ signifies the discrete weight space induced by the quantized codebook $\hat{\mathbf{C}}$.

To facilitate further analysis, we utilize the equivalence between the Frobenius norm and the matrix trace to transform and expand the objective function as:

$$\begin{aligned} \mathcal{L}_{\mathrm{CDC}} = &\ \mathrm{Tr}(\hat{\mathbf{W}}\hat{\mathbf{X}}^\top\hat{\mathbf{X}}\hat{\mathbf{W}}^\top) - 2\,\mathrm{Tr}(\mathbf{W}\mathbf{X}^\top\hat{\mathbf{X}}\hat{\mathbf{W}}^\top) \\ &+ \mathrm{Tr}(\mathbf{W}\mathbf{X}^\top\mathbf{X}\mathbf{W}^\top). \end{aligned} \tag{15}$$

The last term represents the energy of the full-precision output, $\|\mathbf{X}\mathbf{W}^\top\|_F^2$. Since it is independent of the optimization variable $\hat{\mathbf{W}}$, it can be subsumed into the constant term. We define:

$$\mathbf{H} = \hat{\mathbf{X}}^\top\hat{\mathbf{X}}, \ \mathbf{P} = \mathbf{X}^\top\hat{\mathbf{X}}, \tag{16}$$

where $\mathbf{H}$ denotes the Hessian matrix induced by the shifted activations, and $\mathbf{P}$ is the cross correlation matrix between the full-precision and shifted activations. Consequently, the objective function can be rewritten as:

$$\begin{aligned} \mathcal{L}_{\mathrm{CDC}} = &\ \mathrm{Tr}\left(\hat{\mathbf{W}}\mathbf{H}\hat{\mathbf{W}}^\top\right) - 2\,\mathrm{Tr}\left(\mathbf{W}\mathbf{P}\hat{\mathbf{W}}^\top\right) \\ &+ \mathrm{const}. \end{aligned} \tag{17}$$

Directly optimizing this objective within the highly non-convex discrete space $\Omega(\hat{\mathbf{C}})$ is intrinsically an NP-Hard combinatorial optimization problem, lacking a direct analytical solution. To address this, we temporarily relax the discrete codebook constraint. The objective then becomes a convex quadratic function with respect to $\hat{\mathbf{W}}$. Setting its gradient to zero:

$$\frac{\partial \mathcal{L}_{\mathrm{CDC}}}{\partial \hat{\mathbf{W}}} = 2\hat{\mathbf{W}}\mathbf{H} - 2\mathbf{W}\mathbf{P} = \mathbf{0}. \tag{18}$$

Thus, the optimal analytical weight solution in the continuous space is given by:

$$\tilde{\mathbf{W}} = \mathbf{W}\mathbf{P}\mathbf{H}^{-1}. \tag{19}$$

By completing the square, the original discrete optimization objective can be equivalently reformulated as a projection problem centered around $\tilde{\mathbf{W}}$:

$$\begin{aligned} \min_{\hat{\mathbf{W}}} \ &\mathrm{Tr}\left((\hat{\mathbf{W}} - \tilde{\mathbf{W}})\mathbf{H}(\hat{\mathbf{W}} - \tilde{\mathbf{W}})^\top\right) \\ \text{s.t. } &\hat{\mathbf{W}} \in \Omega(\hat{\mathbf{C}}). \end{aligned} \tag{20}$$

This formulation reveals that in the context of joint quantization, directly approximating the original weights $\mathbf{W}$ is insufficient. Instead, $\mathbf{W}$ is first transformed via the cross-distribution mapping $\mathbf{P}\mathbf{H}^{-1}$ to yield a new anchor $\tilde{\mathbf{W}}$. The discrete codebook projection is then performed around this updated anchor. This mechanism more effectively compensates for the distribution shifts induced by the compound compression of WAC.

3.2.2 Causal Error Diffusion and Efficient Update

Due to the discrete constraint $\hat{\mathbf{W}} \in \Omega(\hat{\mathbf{C}})$, the optimization variable $\hat{\mathbf{W}}$ cannot freely approximate the analytical anchor $\tilde{\mathbf{W}}$ in the continuous space. Consequently, each local quantization step introduces perturbations that increase $\mathcal{L}_{\text{CDC}}$. Following GPTVQ (Van Baalen et al., 2024), we perform column-wise quantization and propagate the current quantization error to the remaining unquantized columns in a second-order manner.

Since the objective function is independent across output channels, the problem can be decoupled into optimizing the weight vector $\tilde{\mathbf{w}}$ of a single output channel and its quantized counterpart $\hat{\mathbf{w}}$. When the q-th element is quantized, its local perturbation is defined as:

$$\delta w_q = \hat{w}_q - \tilde{w}_q. \tag{21}$$

To minimize the second order loss increment induced by this perturbation, we seek the optimal perturbation vector $\delta\mathbf{w}$. By introducing a one-hot vector $\mathbf{e}_q$, this can be formulated as the following constrained quadratic programming problem:

$$\min_{\delta\mathbf{w}} \frac{1}{2}\delta\mathbf{w}\mathbf{H}\delta\mathbf{w}^\top, \text{ s.t. } \delta\mathbf{w}\mathbf{e}_q = \hat{w}_q - \tilde{w}_q. \tag{22}$$

Using the method of Lagrange multipliers(Appendix B.1), the optimal analytical solution is derived as:

$$\delta\mathbf{w} = \frac{\hat{w}_q - \tilde{w}_q}{(\mathbf{H}^{-1})_{qq}}(\mathbf{H}^{-1})_{q,:}. \tag{23}$$

Accordingly, we generalize the element-wise compensation derived for a single output channel to all output channels, yielding the matrix-form update for the remaining unquantized columns:

$$\tilde{\mathbf{W}}_{:,q+1:} \leftarrow \tilde{\mathbf{W}}_{:,q+1:} - (\tilde{\mathbf{W}}_{:,q} - \hat{\mathbf{W}}_{:,q})\frac{(\mathbf{H}^{-1})_{q,q+1:}}{(\mathbf{H}^{-1})_{qq}} \tag{24}$$

Rather than explicitly computing and storing $\mathbf{H}^{-1}$, we apply the Cholesky decomposition $\mathbf{H}^{-1} = \mathbf{L}\mathbf{L}^\top$. The update can thus be equivalently reformulated as:

$$\tilde{\mathbf{W}}_{:,q+1:} \leftarrow \tilde{\mathbf{W}}_{:,q+1:} - \frac{\tilde{\mathbf{W}}_{:,q} - \hat{\mathbf{W}}_{:,q}}{L_{qq}}\mathbf{L}^\top_{q,\,q+1:} \tag{25}$$

However, explicitly maintaining the matrix $\tilde{\mathbf{W}}$, which shares the same dimensions as the original weights, incurs significant memory overhead. Based on $\tilde{\mathbf{W}} = \mathbf{W}\mathbf{P}\mathbf{H}^{-1}$, it can be further rewritten as:

$$\tilde{\mathbf{W}} = \mathbf{W} + \mathbf{W}(\mathbf{P} - \mathbf{H})\mathbf{H}^{-1}. \tag{26}$$

Here, we define $\Delta\mathbf{R} = \mathbf{P} - \mathbf{H} = (\mathbf{X} - \hat{\mathbf{X}})^\top\hat{\mathbf{X}}$, which characterizes the cross-distribution residuals induced by multiple quantizations. Equation. (26) reveals the underlying compensation mechanism of CDC: superimposing a correction term, driven by the input distribution shift $(\mathbf{X} - \hat{\mathbf{X}})$, onto the original weights $\mathbf{W}$.

To preserve the causality of column-wise quantization—meaning the quantization error of the current column q can only be compensated by the subsequent unquantized columns ($> q$)—we project the global residual transformation into the Cholesky orthogonal space. By introducing a strictly upper-triangular mask $\mathbf{M}_U$ to filter out all non-causal error backpropagation, we ultimately define the Cross-Distribution Causal Compensation matrix $\mathbf{\Phi}$:

$$\mathbf{\Phi} = ((\Delta\mathbf{R}\mathbf{L}) \odot \mathbf{M}_U)\,\mathbf{L}^\top. \tag{27}$$

Benefiting from this causal design, when quantizing the q-th column, $\mathbf{W}_{:,q}$ in the original memory space has already dynamically absorbed the error diffusion and residual compensation from all previously quantized columns on the left. Thus, it can be directly utilized as the optimal weight for the current step (i.e., $\tilde{\mathbf{W}}_{:,q} \equiv \mathbf{W}_{:,q}$). At this point, the normalized quantization error is defined as $\mathbf{E}_q = \frac{\mathbf{W}_{:,q} - \hat{\mathbf{W}}_{:,q}}{L_{qq}}$. Finally, without the need to explicitly maintain $\tilde{\mathbf{W}}$, the weight update formula can be streamlined to:

$$\mathbf{W}_{:,q+1:} \leftarrow \mathbf{W}_{:,q+1:} - \mathbf{E}_q\mathbf{L}_{q,q+1:} + \mathbf{W}_{:,q}\mathbf{\Phi}_{q,q+1:}. \tag{28}$$

This update seamlessly integrates second-order error diffusion and cross-distribution residual compensation. While strictly maintaining the causality

of column-wise quantization, it effectively alleviates the error accumulation problem inherent in the joint quantization of weights, activations, and codebooks.

4 Experiments

4.1 Experimental Settings

Models and Datasets. We conduct extensive experiments on several mainstream large language models, LLaMA2 (7B/13B) (Touvron et al., 2023), LLaMA3 (8B) (Grattafiori et al., 2024), and LLaMA3.2 (1B/3B) (Meta AI, 2024). We report perplexity (PPL) on WikiText-2 (Merity et al., 2016) and evaluate zero-shot accuracy on six commonsense reasoning tasks: PiQA (Bisk et al., 2020), Arc-Easy, Arc-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), and BoolQ (Clark et al., 2019), using the lm-eval framework (Gao et al., 2024) to report the average accuracy.

Baselines and Implementation Details. To evaluate TriVQ, we compare it with representative quantization baselines, including equivalent-transformation methods (OmniQuant (Shao et al., 2024) and DuQuant (Lin et al., 2024)) and rotation-based methods (QuaRot (Ashkboos et al., 2024) and SpinQuant (Liu et al., 2025a)). For rotation-based baselines, we further evaluate GPTQ (Frantar et al., 2022) and GPTAQ (Li et al., 2025) as weight optimization strategies. Experiments are conducted under W4A4, W3A4, and W2A4 settings. Activations are symmetrically quantized per token, while scalar-quantized weights are symmetrically quantized per channel. For TriVQ, we apply Hadamard rotation (Halko et al., 2011) for activation outlier suppression and use per-tensor asymmetric quantization for the codebook. Calibration uses 128 randomly sampled WikiText-2 training sequences with a length of 2048 tokens. All experiments are run on a single NVIDIA A100 40GB GPU.

4.2 Main Results

Language Generation Tasks. As shown in Table 1, at W4A4, TriVQ achieves highly competitive performance but marginally trails the state-of-the-art on certain models (e.g., 6.04 vs. 5.85 for QuaRot+GPTAQ on LLaMA2-7B). Because 4-bit scalar quantization adequately fits the weight distribution, TriVQ's additional codebook quantization error partially offsets its structural advantages. However, as bit-widths drop to W3A4 and W2A4, the drastically reduced representation space causes severe baseline degradation, allowing TriVQ to demonstrate exceptional robustness. Under the extreme W2A4 configuration, baselines suffer catastrophic collapses. On LLaMA2-7B, OmniQuant's PPL surges to 1.25×10^5 and SpinQuant+GPTAQ degrades to 56.59, while TriVQ robustly maintains 9.32. This dominance persists across LLaMA2-13B (yielding 7.78 vs. the second-best 9.52) and the LLaMA3.2 1B/3B models (achieving 36.63 and 18.14, respectively). This proves that the synergistic integration of weighted codebook quantization and cross-distribution calibration effectively mitigates the severe distribution shifts induced by extreme low-bit compression.

Zero-Shot Tasks. Zero-shot reasoning results (Table 1) corroborate our language generation findings. At W4A4, TriVQ performs comparably to optimal baselines, but establishes clear superiority at W3A4 (e.g., 64.9% on LLaMA2-7B). Under extreme W2A4 constraints, TriVQ achieves comprehensive dominance. While most baselines severely degrade on LLaMA2-7B (e.g., SpinQuant+GPTAQ drops to 45.3%), TriVQ robustly maintains 53.4%. Similarly, it significantly outperforms all competitors on the challenging LLaMA3-8B (51.2%) and LLaMA3.2-3B (48.5%) models. This demonstrates that TriVQ maximizes the retention of full-precision semantic and reasoning capabilities in ultra-low-bit scenarios.

4.3 Ablation Studies

Effectiveness of Weighted Codebook Quantization. Compressing high-precision clustered codebooks to ultra-low bit-widths inevitably introduces secondary quantization errors. Thus, we compare various parameter calibration and rounding strategies under per-tensor codebook quantization, utilizing the uncompressed codebook (W2A4C16) as the baseline. As shown in Table 2, directly compressing the codebook to 4-bit via the standard MinMax method causes significant performance degradation, increasing the PPL to 18.62. While an MSE-based grid search slightly improves the PPL to 18.40, it ignores the unequal impact of codebook errors during weight reconstruction. By introducing our proposed Dual-Weighted Loss (DWL), the PPL further drops to 18.28. Integrating adaptive soft rounding (AdaRound) upon DWL achieves the optimal PPL of 18.14 under the W2A4C4 configuration. Notably, its zero-shot accuracy (48.5%) marginally surpasses even the 16-bit codebook

Bits	Method	LLaMA2 7B		LLaMA2 13B		LLaMA3 8B		LLaMA3.2 1B		LLaMA3.2 3B	
		Wiki PPL (↓)	Zero-shot[6] Avg. (↑)	Wiki PPL (↓)	Zero-shot[6] Avg. (↑)	Wiki PPL (↓)	Zero-shot[6] Avg. (↑)	Wiki PPL (↓)	Zero-shot[6] Avg. (↑)	Wiki PPL (↓)	Zero-shot[6] Avg. (↑)
FP16	-	5.47	70.5	4.88	73.3	6.44	74.3	9.89	60.1	7.81	68.6
W4A4 (C4)	OmniQuant	14.60	49.9	12.30	53.1	2.43×10^2	38.6	3.83×10^2	38.9	1.58×10^2	39.2
	DuQuant	6.08	61.0	5.33	63.8	8.56	65.8	-	-	-	-
	QuaRot+GPTQ	6.00	67.2	5.30	70.6	7.80	67.1	13.48	54.1	9.75	60.9
	QuaRot+GPTAQ	5.85	68.1	5.17	71.2	7.36	69.6	13.15	54.7	9.57	61.7
	SpinQuant+GPTQ	5.90	67.1	5.20	70.7	7.26	69.8	15.30	53.5	9.16	62.1
	SpinQuant+GPTAQ	5.85	68.0	5.19	71.5	7.19	71.3	12.67	54.8	9.01	61.6
	TriVQ	6.04	66.5	5.26	70.9	7.74	68.6	13.83	54.4	9.71	60.4
W3A4 (C4)	OmniQuant	1.41×10^2	39.4	83.38	43.9	4.43×10^3	35.6	1.45×10^3	36.8	1.32×10^3	36.2
	DuQuant	6.86	63.5	5.85	66.5	21.61	46.2	-	-	-	-
	QuaRot+GPTQ	6.75	63.7	5.80	68.9	9.14	65.7	17.97	50.0	11.68	57.9
	QuaRot+GPTAQ	6.44	64.3	5.63	68.1	8.54	65.2	16.10	50.5	10.93	58.6
	SpinQuant+GPTQ	6.61	64.8	5.68	69.4	8.47	66.7	17.14	49.5	10.85	58.7
	SpinQuant+GPTAQ	6.30	64.4	5.49	69.5	8.06	67.5	15.43	51.2	10.24	60.7
	TriVQ	6.32	64.9	5.50	69.1	8.34	67.4	15.48	51.5	10.64	59.8
W2A4 (C4)	OmniQuant	1.25×10^5	36.7	4.73×10^4	36.6	5.19×10^4	36.8	1.56×10^4	36.5	1.07×10^4	36.3
	DuQuant	3.86×10^2	36.0	14.24	47.5	4.90×10^4	35.9	-	-	-	-
	QuaRot+GPTQ	53.85	37.9	16.14	46.6	55.32	41.7	1.06×10^2	39.2	53.18	39.0
	QuaRot+GPTAQ	22.64	44.3	9.87	48.6	24.32	44.7	73.65	38.9	29.53	42.8
	SpinQuant+GPTQ	68.02	40.5	17.15	44.1	58.16	41.9	1.03×10^2	39.8	47.79	39.4
	SpinQuant+GPTAQ	56.59	45.3	9.52	49.5	21.34	45.8	92.82	40.6	24.00	43.8
	TriVQ	9.32	53.4	7.78	56.6	14.07	51.2	36.63	41.5	18.14	48.5

Table 1: Performance comparison of various quantization methods across different LLaMA model families on language generation (WikiText-2 PPL) and zero-shot reasoning tasks. All TriVQ experiments use 4-bit codebooks.

baseline (48.3%). These results effectively validate the critical roles of both DWL and AdaRound.

Quantization Strategy	Precision	PPL (↓)	Avg.[6] (↑)
Baseline	W2A4C16	17.93	48.3
MinMax	W2A4C4	18.62	46.6
MSE	W2A4C4	18.40	47.3
DWL	W2A4C4	18.28	46.1
DWL + AdaRound	W2A4C4	**18.14**	**48.5**

Table 2: Ablation study of different codebook quantization strategies on Wiki and six zero-shot datasets.

Effectiveness of Cross-Distribution Calibration. To evaluate the Cross-Distribution Calibration (CDC) mechanism, we ablate its error compensation sources (Table 3). At W2A4C4, optimizing solely for weight discretization (Only Weight) yields severe degradation (38.48 PPL), which reveals the failure of the identical-distribution assumption and highlights the progressive impact of coupled errors. Introducing dual compensation to inversely absorb shifts from either activations (W + A) or codebooks (W + C) significantly improves the PPL to 22.76 and 23.59, respectively. Ultimately, the full CDC (W + A + C) jointly models the triple-compression errors, analytically projecting them into a causal compensation matrix. This global absorption of all antecedent errors achieves the optimal 18.14 PPL and 48.5% zero-shot accuracy, confirming the essential role of cross-distribution calibration.

Compensated Error	Precision	PPL (↓)	Avg.[6] (↑)
Only Weight	W2A4C4	38.48	42.5
Weight + Activation	W2A4C4	22.76	45.8
Weight + Codebook	W2A4C4	23.59	45.7
Full CDC (W + A + C)	W2A4C4	**18.14**	**48.5**

Table 3: Ablation study of error compensation sources within the CDC on Wiki and six zero-shot datasets.

5 Conclusion

In this paper, we introduced TriVQ, a framework for end-to-end integer-only inference via joint quantization of weights, activations, and codebooks (WAC). TriVQ overcomes codebook quantization trade-offs using a Dual-Weighted Loss (DWL) with soft-rounding, and mitigates compounding errors across layers via Cross-Distribution Calibration (CDC). Evaluations on large language models confirm that TriVQ delivers unparalleled robustness and performance under extreme ultra-low-bit configurations (e.g., W2A4), fundamentally preventing the performance collapse observed in state-of-the-art baselines. By seamlessly translating VQ-based compression into practical computational speedups, TriVQ offers a highly effective and hardware-friendly pathway for the scalable deployment of LLMs.

Limitations

Constrained by available computational resources, our current empirical evaluations are limited to models with up to 13 billion parameters. Consequently, we have yet to validate TriVQ on massively large-scale models (e.g., 70B variants) or explore diverse architectures like Mixture-of-Experts (MoE). Future work will aim to secure the necessary resources to comprehensively evaluate the scalability of our joint quantization framework across these broader model families.

While our efficiency analysis demonstrates substantial hardware-friendly acceleration, the evaluation currently focuses on kernel-level execution latency for isolated GEMM operations. We have not yet assessed the end-to-end generation throughput within fully integrated serving frameworks, such as vLLM or TensorRT-LLM. Deploying TriVQ into a production-level engine to benchmark complete system performance during autoregressive decoding remains an important direction for subsequent research.

Additionally, our proposed method introduces additional computational overhead during the offline post-training calibration phase. Compared to traditional vector quantization baselines (e.g., GPTVQ and VPTQ), TriVQ's rigorous iterative optimization—specifically DWL fine-tuning and CDC—requires more time before deployment. Although these mechanisms are crucial for mitigating compounding errors in extreme ultra-low-bit settings, optimizing calibration efficiency and exploring faster convergence strategies will be prioritized in future iterations.

In this paper, we only use AI tools for grammar checking and code completion.

References

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. 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)*, pages 2924–2936.

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

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35:30318–30332.

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

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

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE.

Yuhang Li, Ruokai Yin, Donghyun Lee, Shiting Xiao, and Priyadarshini Panda. 2025. Gptaq: Efficient finetuning-free quantization for asymmetric calibration. *arXiv preprint arXiv:2504.02692*.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. 2024. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. *Advances in Neural Information Processing Systems*, 37:87766–87800.

Yifei Liu, Jicheng Wen, Yang Wang, Shengyu Ye, Li Lyna Zhang, Ting Cao, Cheng Li, and Mao Yang. 2024. Vptq: Extreme low-bit vector post-training quantization for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8181–8196.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2025a. Spinquant: Llm quantization with learned rotations. In *International Conference on Learning Representations*, volume 2025, pages 92009–92032.

Zihan Liu, Xinhao Luo, Junxian Guo, Wentao Ni, Yangjie Zhou, Yue Guan, Cong Guo, Weihao Cui, Yu Feng, Minyi Guo, and 1 others. 2025b. Vq-llm: High-performance code generation for vector quantization augmented llm inference. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1496–1509. IEEE.

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

Meta AI. 2024. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. Meta AI Blog. Accessed: 2026-05-25.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. 2020. Up or down? adaptive rounding for post-training quantization. In *International conference on machine learning*, pages 7197–7206. PMLR.

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

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Gao Peng, Yu Qiao, and Ping Luo. 2024. Omniquant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, volume 2024, pages 45472–45496.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single GPU. In *International Conference on Machine Learning (ICML)*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Mart Van Baalen, Andrey Kuzmin, Ivan Koryakovskiy, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. 2024. Gptvq: The blessing of dimensionality for llm quantization. *arXiv preprint arXiv:2402.15319*.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. Bitnet: Scaling 1-bit transformers for large language models. *Preprint*, arXiv:2310.11453.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in neural information processing systems*, 35:27168–27183.

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

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. 2022. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113.

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

A Inference Acceleration Analysis

Unlike previous vector quantization (VQ) methods that primarily focus on extreme weight compression, TriVQ concurrently compresses weights, activations, and codebooks under the W2A4C4 configuration. This triple-compression paradigm significantly mitigates memory traffic bottlenecks during General Matrix Multiply (GEMM) operations. Furthermore, the specialized storage format of the codebooks and activations enables the direct utilization of hardware-intrinsic fixed-point multiply-accumulate (MAC) instructions.

To empirically validate the acceleration, we developed a custom CUDA kernel for TriVQ and

benchmarked it against the highly optimized CUTLASS FP16 baseline. Under the configuration of $m = 15$, $k = 4096$, and $n = 4096$, the FP16 kernel requires $54.37\,\mu$s, whereas TriVQ takes only $27.74\,\mu$s. When scaling to a larger weight matrix ($n = 11408$), the memory wall becomes more pronounced: the latency of FP16 surges to $198.80\,\mu$s, while TriVQ completes the execution in just $52.29\,\mu$s, achieving a remarkable speedup of approximately $3.8\times$. These results substantiate that beyond theoretical performance retention, TriVQ is inherently hardware-friendly, successfully translating structural compression into substantial practical inference acceleration.

B Theoretical Derivation

B.1 Derivation of the Optimal Solution via Lagrange Multipliers

As formulated in the main text, to minimize the second-order loss increment for a single output channel, we seek the optimal perturbation row vector $\delta\mathbf{w}$ by solving the following constrained quadratic programming problem:

$$\begin{aligned} \min_{\delta\mathbf{w}} \quad & \frac{1}{2}\delta\mathbf{w}\mathbf{H}\delta\mathbf{w}^\top \\ \text{s.t.} \quad & \delta\mathbf{w}\mathbf{e}_q = \hat{w}_q - \tilde{w}_q. \end{aligned} \tag{29}$$

To solve this, we introduce the Lagrange multiplier λ and construct the Lagrangian function $\mathcal{L}(\delta\mathbf{w}, \lambda)$:

$$\mathcal{L}(\delta\mathbf{w}, \lambda) = \frac{1}{2}\delta\mathbf{w}\mathbf{H}\delta\mathbf{w}^\top - \lambda(\delta\mathbf{w}\mathbf{e}_q - (\hat{w}_q - \tilde{w}_q)). \tag{30}$$

Setting the partial derivative of $\mathcal{L}$ with respect to the row vector $\delta\mathbf{w}$ to zero yields:

$$\frac{\partial\mathcal{L}}{\partial\delta\mathbf{w}} = \delta\mathbf{w}\mathbf{H} - \lambda\mathbf{e}_q^\top = \mathbf{0} \implies \delta\mathbf{w} = \lambda\mathbf{e}_q^\top\mathbf{H}^{-1}. \tag{31}$$

Next, we substitute this expression back into the equality constraint $\delta\mathbf{w}\mathbf{e}_q = \hat{w}_q - \tilde{w}_q$ to determine the multiplier λ:

$$(\lambda\mathbf{e}_q^\top\mathbf{H}^{-1})\mathbf{e}_q = \hat{w}_q - \tilde{w}_q. \tag{32}$$

Note that the quadratic form $\mathbf{e}_q^\top\mathbf{H}^{-1}\mathbf{e}_q$ mathematically extracts the scalar diagonal element at the q-th row and q-th column of the inverse Hessian matrix, denoted as $(\mathbf{H}^{-1})_{qq}$. Thus, the equation simplifies to:

$$\lambda(\mathbf{H}^{-1})_{qq} = \hat{w}_q - \tilde{w}_q \implies \lambda = \frac{\hat{w}_q - \tilde{w}_q}{(\mathbf{H}^{-1})_{qq}}. \tag{33}$$

Finally, substituting the derived λ back into Equation (31), and recognizing that right-multiplying $\mathbf{e}_q^\top$ by $\mathbf{H}^{-1}$ extracts the q-th row of the matrix, denoted as $(\mathbf{H}^{-1})_{q,:}$, we obtain the closed-form analytical solution:

$$\delta\mathbf{w} = \frac{\hat{w}_q - \tilde{w}_q}{(\mathbf{H}^{-1})_{qq}}(\mathbf{H}^{-1})_{q,:}. \tag{34}$$

C Algorithmic Details

The detailed execution flow of our proposed TriVQ framework is formally described in the following algorithms. Algorithm 1 outlines the Dual-Weighted Codebook Quantization (DWL) procedure. Operating on an initialized floating-point codebook, it iteratively optimizes the continuous clipping bounds and soft-rounding offsets using a Straight-Through Estimator (STE). The optimization is explicitly driven by a dual-weighted objective ($\mathbf{H}_C$) that accounts for both the codebook usage frequency and the input activation sensitivity.

Algorithm 1 DWL: Dual-Weighted Codebook Quantization

Input: FP Codebook $\mathbf{C}$, Activation Hessian $\mathbf{H}_X$, One-Hot Assignment matrix $\mathbf{Z}$, Bit-width b, Epochs E
Output: Quantized codebook $\hat{\mathbf{C}}$, Optimal params $\theta^* = \{\gamma^*, \beta^*, \mathbf{V}^*\}$
Initialize clipping bounds γ, β; soft-rounding offsets $\mathbf{V} \leftarrow \mathbf{0}$
$\mathbf{F} \leftarrow \mathbf{Z}\mathbf{Z}^\top$ ▷ Codebook usage frequency
$\mathbf{H}_C \leftarrow \mathbf{F} \otimes \mathbf{H}_X$ ▷ Dual-weighted matrix, via Eq. (12)
for $epoch = 1, 2, \ldots, E$ **do**
  $s \leftarrow \frac{\beta-\gamma}{2^b-1}, \quad z \leftarrow -\frac{\gamma}{s}$
  $\mathbf{C}_{\text{int}} \leftarrow \text{quant}(\mathbf{C}, \mathbf{V}, s, z)$ ▷ Using Eq. (13)
  $\hat{\mathbf{C}} \leftarrow s(\mathbf{C}_{\text{int}} - z)$
  $\Delta\mathbf{C} \leftarrow \hat{\mathbf{C}} - \mathbf{C}$
  $\mathcal{L}_{\text{DWL}} \leftarrow \text{vec}(\Delta\mathbf{C}^\top)^\top\mathbf{H}_C\text{vec}(\Delta\mathbf{C}^\top)$ ▷ Evaluate objective
  Estimate gradients $\nabla_\theta\mathcal{L}_{\text{DWL}}$ using STE for $\lfloor\cdot\rfloor$
  Update parameters $\theta = \{\gamma, \beta, \mathbf{V}\}$ via Adam optimizer
end for
Compute the final scale s^* and integer zero-point z^* to reconstruct $\hat{\mathbf{C}}$
Return: $\hat{\mathbf{C}}$ and θ^*

Building upon this core quantization module,

Algorithm 2 presents the comprehensive Cross-Distribution Calibration (CDC) pipeline. To ensure computational tractability and memory efficiency, CDC processes the weight matrix in a structured block-wise and group-wise manner. For each group, the weights are reshaped for vectorization, clustered via Hessian-weighted K-Means, and subsequently quantized by invoking Algorithm 1. Once the local weights are reconstructed, the resulting quantization errors are causally distributed to the remaining unquantized columns. This step seamlessly integrates standard second-order error diffusion with our proposed cross-distribution residual compensation ($\mathbf{\Phi}$), effectively mitigating error accumulation at both the intra-block and inter-block levels.

Algorithm 2 CDC: Joint W-A-C Quantization with Group-wise VQ

Input: FP weights $\mathbf{W}$, Original acts $\mathbf{X}$, Shifted acts $\hat{\mathbf{X}}$, Block size B, Group size g, Vector dim d, Args for DW_Quant ▷ Note: B is a multiple of g, g is a multiple of d
Output: Integer codebooks $\{\hat{\mathbf{C}}\}$, Assignment matrices $\{\mathbf{I}\}$, Quantized weights $\hat{\mathbf{W}}$
$\mathbf{H} \leftarrow \hat{\mathbf{X}}^\top \hat{\mathbf{X}}$
$\mathbf{L} \leftarrow \mathit{Cholesky}(\mathbf{H}^{-1})$ ▷ $\mathbf{L}$ is lower triangular
$\Delta\mathbf{R} \leftarrow (\mathbf{X} - \hat{\mathbf{X}})^\top \hat{\mathbf{X}}$ ▷ Cross-Distribution Residual
$\mathbf{\Phi} \leftarrow ((\Delta\mathbf{R}\mathbf{L}) \odot \mathbf{M}_U)\, \mathbf{L}^\top$ ▷ Causal Compensation Matrix
Initialize $\hat{\mathbf{W}} \leftarrow \mathbf{0}$, $\mathbf{E} \leftarrow \mathbf{0}$
for $i = 0, B, 2B, \ldots, ic - B$ **do**
  for $j = i, i + d, \ldots, i + B - d$ **do**
    if $j \bmod g = 0$ **then**
      shape $= [oc \times (g/d), d]$
      $\mathbf{W}_{\text{vec}} \leftarrow \text{Vec}(\mathbf{W}_{:j,j+g-1},\ \text{shape})$
      $\mathbf{H}_g \leftarrow \text{diag}(\mathbf{H}_{j:j+g-1})$
      $\mathbf{C}, \mathbf{Z}_{\text{vec}} \leftarrow \text{KMeans}(\mathbf{W}_{\text{vec}},\ \mathbf{H}_g)$
      $\hat{\mathbf{C}} \leftarrow \text{DW_Quant}(\mathbf{C}, \mathbf{Z}_{\text{vec}}, \mathbf{H}_g, \text{Args})$
    end if
    $q = j, \ldots, j + d - 1$
    $\hat{\mathbf{W}}_{:,q},\ \mathbf{I} \leftarrow \text{vq_quantize}(\mathbf{W}_{:,q},\ \hat{\mathbf{C}})$
    $\mathbf{e}_q \leftarrow (\mathbf{W}_{:,q} - \hat{\mathbf{W}}_{:,q})/\mathbf{L}_{qq}$
    $\mathbf{E}_{:,q} \leftarrow \mathbf{e}_q$
    $\mathbf{W}_{:,q+1:} \leftarrow \mathbf{W}_{:,q+1:} - \mathbf{e}_q \mathbf{L}_{q,q+1:} + \mathbf{W}_{:,q} \mathbf{\Phi}_{q,q+1:}$
  end for
  $b = i \ldots i + B - 1$
  $r = i + B \ldots ic - 1$
  $\mathbf{W}_{:,R} \leftarrow \mathbf{W}_{:,R} - \mathbf{E}_{:,b} \mathbf{L}_{b,R} + \mathbf{W}_{:,b} \mathbf{\Phi}_{b,R}$
end for
Return: $\{\hat{\mathbf{C}}\}, \{\mathbf{I}\}, \hat{\mathbf{W}}$