TERNOVA: ADAPTIVE LOW-RANK POST-TRAINING TERNARIZATION FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have achieved remarkable capabilities, yet their massive memory footprint and inference cost hinder efficient deployment. Ternary post-training quantization (PTQ) offers an attractive solution by combining compact storage with richer representation capacity than binary quantization. However, existing ternary PTQ methods remain challenged by limited ternary representation capacity, heterogeneous structural sensitivity across Transformer modules, and the mismatch between local reconstruction and model-level objectives. To address these challenges, we propose Ternova, an adaptive low-rank post-training ternarization framework for LLMs. Ternova is built on three core components: 1) Hessian-Aware Annealed Ternary Factorization (HATF) constructs high-fidelity low-rank ternary representations through sensitivity-aware factorization and annealed discrete refinement; 2) Sensitivity-Reweighted Mixed-Rank Allocation (SRMA) adaptively distributes rank capacity according to structural irreplaceability, task sensitivity, and storage cost; and 3) Task-Aligned Ternary Refinement (TATR) further aligns the compressed model with task-level behavior through global scale calibration and sparse topology refinement. Ternova preserves a compact ternary deployment form without additional dense high-precision branches. Extensive experiments across diverse LLM families demonstrate that Ternova achieves competitive or superior performance over existing low-bit PTQ methods at only 1.59 BPW while retaining competitive reasoning capabilities. On Qwen3.5-27B, Ternova achieves 74.62% average zero-shot accuracy, approaching the 75.97% BF16 baseline. Moreover, Ternova achieves roughly 3× inference speedup and up to 80.6% memory savings over BF16, demonstrating a strong balance among accuracy, compression, and practical efficiency.

1 INTRODUCTION

Transformer-based large language models (LLMs) have achieved remarkable progress in language understanding, code generation, mathematical reasoning, and agentic tasks, while their capabilities continue to benefit from increasing model scale. This trend appears in both dense and mixture-of-experts (MoE) architectures. Llama-3.1 (Grattafiori et al., 2024) scales dense Transformers to 405B parameters, while recent MoE models further expand total model capacity: Qwen3.5-397B-A17B (Qwen Team, 2026a) contains 397B parameters, GLM-5.2 GLM-5-Team et al. (2026) reaches 744B, and DeepSeek-V4-Pro (DeepSeek-AI, 2026) scales to 1.6T parameters. Although MoE reduces per-token computation by activating only a subset of parameters, the rapidly growing total parameter count still

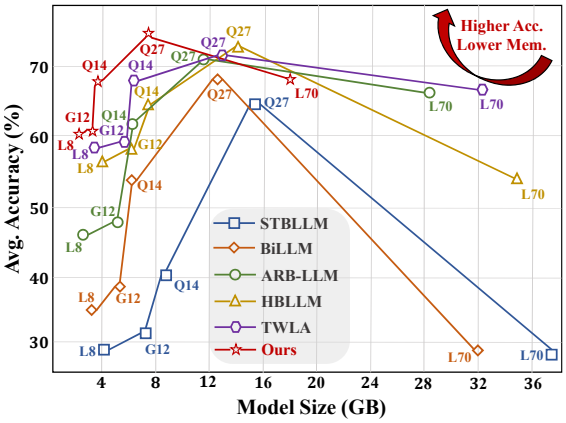


Figure 1: Performance–memory trade-off across LLMs. Our method achieves higher average accuracy with lower memory cost than existing methods.

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.

imposes substantial storage costs. Dense models face an even more direct memory bottleneck. For example, storing a 400B-parameter dense model in BF16 requires approximately 800 GB for model weights alone. Therefore, reducing the memory footprint and inference cost of LLMs while preserving their capabilities is critical for practical deployment.

Post-training quantization (PTQ) has become an important paradigm for LLM compression because it reduces storage and inference costs without costly retraining. Recent studies have pushed PTQ toward ultra-low-bit binary and ternary representations. Binary quantization offers highly compact storage and simple computation, while ternarization introduces an additional zero state for stronger expressive capacity with a similarly simple discrete representation. However, aggressive quantization often causes severe accuracy degradation. Existing binary and ternary PTQ methods (Dong et al., 2024; Huang et al., 2024; Li et al., 2024; Chen et al., 2025; Zhao et al., 2026) commonly recover performance through fine-grained scales, group-wise parameters, auxiliary compensation structures, or other metadata. As shown in Fig. 1, these auxiliary components can substantially increase practical memory cost beyond the nominal low-bit representation. In contrast, quantization-aware training (QAT) methods (Ma et al., 2024; Chen et al., 2024) can achieve higher-quality ultra-low-bit models but require large-scale training data and substantial optimization cost. These limitations motivate ternary PTQ as an attractive balance among compactness, representation capacity, and computational simplicity. Nevertheless, achieving high accuracy with genuinely low memory cost and limited auxiliary overhead remains challenging.

We examine this challenge from three perspectives: local reconstruction, structural allocation, and global optimization. ❶ **High-fidelity ternary reconstruction is intrinsically difficult.** A common formulation approximates a pretrained weight matrix as $W \approx \alpha Q$, where $Q \in \{-1, 0, +1\}^{m \times n}$. A single ternary matrix has limited capacity to capture the anisotropic and channel-heterogeneous distributions of LLM weights. Moreover, hard projection or local discrete optimization may prematurely fix incorrect zero and sign assignments, making accurate reconstruction difficult under limited calibration budgets. ❷ **Transformer modules exhibit heterogeneous compression sensitivity and structural redundancy.** Different layers and projection matrices respond differently to quantization perturbations. Uniform rank or compression configurations may therefore over-compress sensitive modules while wasting capacity on redundant ones. ❸ **Local reconstruction is not fully aligned with global task performance.** Weight- or layer-level reconstruction errors are only proxies for the final language modeling objective, and small local errors may accumulate through deep Transformer blocks. Effective ternary PTQ therefore requires stronger local representations, sensitivity-aware capacity allocation, and task-driven global calibration.

To address these challenges, we propose **Ternova**, an adaptive low-rank post-training ternarization framework that jointly improves ternary representation, capacity allocation, and global task alignment. ❶ **Hessian-Aware Annealed Ternary Factorization (HATF)** improves local reconstruction by constructing a Hessian-aware surrogate space and replacing single-matrix ternarization with dual-ternary low-rank factorization. Annealed refinement further improves zero and sign assignments while separating discrete structure search from continuous scale fitting. ❷ **Sensitivity-Reweighted Mixed-Rank Allocation (SRMA)** addresses heterogeneous redundancy and sensitivity by combining refit-aware rank-component importance, Fisher-style matrix sensitivity, and storage cost to allocate ranks under a fixed BPW budget. ❸ **Task-Aligned Ternary Refinement (TATR)** bridges local reconstruction and model-level objectives through end-to-end distillation, globally recalibrating continuous scales and selectively refining ternary states. Ternova retains a compact low-rank ternary form at deployment without introducing additional dense high-precision weight branches.

Our key contributions are summarized as follows:

- We identify three key challenges in post-training ternarization of LLMs: limited ternary representation capacity, heterogeneous compression sensitivity and structural redundancy, and the mismatch between local reconstruction and model-level behavior.
- We propose **Ternova**, an adaptive low-rank ternary PTQ framework combining **HATF** for high-fidelity ternary factorization, **SRMA** for sensitivity- and cost-aware mixed-rank allocation, and **TATR** for task-aligned global refinement.
- Extensive experiments demonstrate that Ternova achieves a favorable accuracy–memory trade-off over existing low-bit PTQ methods at only 1.59 BPW, while retaining competitive reasoning performance and delivering practical memory savings and inference acceleration.

2 Related Work

2.1 Post-Training Quantization for Large Language Models

Post-training quantization (PTQ) has become a practical approach for compressing LLMs without costly retraining. Existing methods improve low-bit robustness from different perspectives. GPTQ (Frantar et al., 2022) exploits second-order information for error compensation, while AWQ (Lin et al., 2024) and OWQ (Lee et al., 2024) preserve important or outlier weights using activation-aware statistics. Distribution-shaping methods further improve quantization friendliness: SmoothQuant (Xiao et al., 2024) redistributes activation outliers through scaling, QuaRot (Ashkboos et al., 2024) and SpinQuant (Liu et al., 2025) apply rotation transformations, and FlatQuant (Sun et al., 2025) learns layer-wise affine transformations. For more aggressive compression, QuIP# (Tseng et al., 2024) combines incoherence processing with lattice-based codebooks, while recent binary PTQ methods such as BiLLM (Huang et al., 2024), ARB-LLM (Li et al., 2024), and HBLLM (Chen et al., 2025) push pretrained LLMs toward 1-bit representations. Our work follows this ultra-low-bit PTQ direction and focuses on post-training ternarization.

2.2 Ternary Quantization for Large Language Models

Ternary quantization provides richer representations than binary quantization while retaining a discrete form. Early works such as TWN (Li et al., 2022) and TTQ (Zhu et al., 2017) explored ternary projection and joint optimization of ternary assignments and scaling factors. TernaryBERT (Zhang et al., 2020) later combined ternary compression with knowledge distillation, while TerDiT (Lu et al., 2025) extended ternary representations to diffusion Transformers. For LLMs, TernaryLLM (Chen et al., 2024) and BitNet b1.58 (Ma et al., 2024) demonstrated ternary performance through training-based optimization, but require modifying or retraining the model. Recent ternary PTQ methods, including PT^2-LLM (Yan et al., 2026), PTQTP (Xiao et al., 2025), and TWLA (Zhao et al., 2026), avoid retraining by improving thresholding, scaling, or local compensation. Nevertheless, these methods rely on single ternary matrix representations with auxiliary parameters, leaving representation capacity and effective-BPW efficiency as key limitations.

3 Method

Overview. To address the challenges identified in Sec. 1, we propose Ternova, a three-stage framework for accurate and efficient post-training ternarization of LLMs (Fig. 2). Hessian-Aware Annealed Ternary Factorization (HATF) constructs high-fidelity ternary low-rank representations (Sec. 3.1), Sensitivity-Reweighted Mixed-Rank Allocation (SRMA) adaptively allocates rank capacity under a global BPW budget (Sec. 3.2), and Task-Aligned Ternary Refinement (TATR) aligns the compressed model with task-level behavior through scale calibration and sparse topology refinement (Sec. 3.3). Detailed pseudocode is provided in Appendix A.1.

3.1 Hessian-Aware Annealed Ternary Factorization (HATF)

Given a pretrained linear layer weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, HATF constructs a high-fidelity ternary low-rank representation through three stages: Hessian-aware preconditioning, TSVID-guided ADMM factorization, and annealed joint refinement. The surrogate form is

$$\widehat{W}' = \text{Diag}(a) T_A \,\text{Diag}(m) T_B \,\text{Diag}(b), \tag{1}$$

where T_A and T_B are ternary factors with entries in $\{-1, 0, +1\}$, while a, m, and b are continuous scaling vectors that capture output-side, latent-rank, and input-side magnitude variations.

3.1.1 Stage I: Hessian-Aware Preconditioning

Conventional Frobenius reconstruction treats all weight perturbations equally and ignores the non-uniform sensitivity of different input and output channels. To incorporate task-dependent sensitivity, we follow second-order compression methods (Boža & Macko, 2025) and approximate the local curvature using K-FAC (Martens & Grosse, 2015). For a linear transformation $y = Wx$, with input

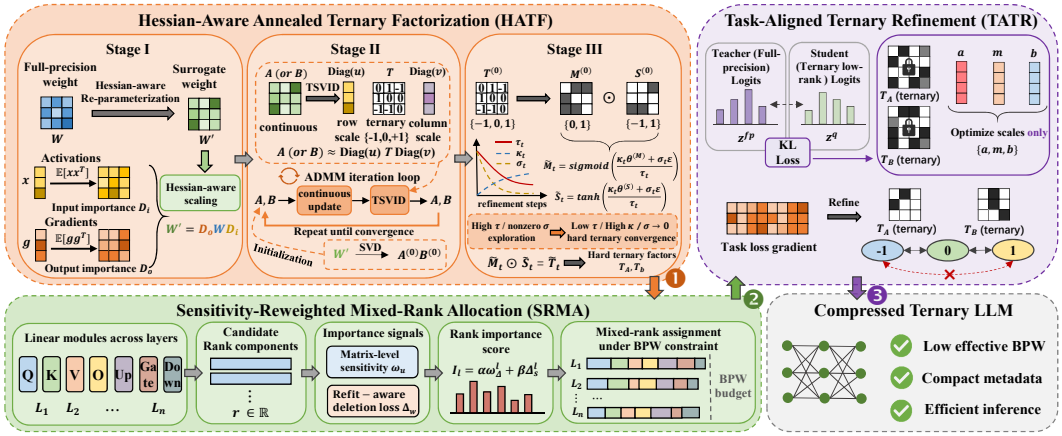


Figure 2: Overview of Ternova. **HATF** improves ternary representation, **SRMA** allocates rank capacity under a global BPW budget, and **TATR** aligns the compressed model with task-level behavior.

activation x and output gradient $g = \partial \mathcal{L}/\partial y$,

$$H_W \approx \mathbb{E}[xx^T] \otimes \mathbb{E}[gg^T]. \tag{2}$$

Rather than maintaining the full curvature factors, we retain their diagonal statistics and define $i_j = \sqrt{\mathbb{E}[x_j^2]}$ and $o_u = \sqrt{\mathbb{E}[g_u^2]}$, with $D_i = \mathrm{Diag}(i)$ and $D_o = \mathrm{Diag}(o)$, yielding

$$\mathcal{L}_{\mathrm{rec}} \approx \left\| D_o(W - \widehat{W}) D_i \right\|_F^2 = \left\| W' - \widehat{W}' \right\|_F^2, \tag{3}$$

where $W' = D_o W D_i$ and $\widehat{W}' = D_o \widehat{W} D_i$. Thus, curvature information is encoded once into the surrogate space, allowing all subsequent factorization and refinement to share the same sensitivity-aware geometry. After optimization, the original-space approximation is recovered as $\widehat{W} = D_o^{-1} \widehat{W}' D_i^{-1}$, where the diagonal transformations are absorbed into the final scaling vectors without additional inference overhead. Detailed derivations are provided in Appendix A.2.1.

3.1.2 Stage II: TSVID-Guided ADMM Ternary Factorization

Given the Hessian-aware surrogate matrix W', we seek a compact factorized ternary representation with stronger capacity than direct single-matrix ternarization. Jointly optimizing discrete ternary states and continuous scales, however, creates a large combinatorial search space. We therefore first construct a deterministic low-error initialization from the optimal rank-r continuous approximation $W'_r = U_r \Sigma_r V_r^T$ (Mirsky, 1960):

$$A^{(0)} = U_r \Sigma_r^{1/2}, \qquad B^{(0)} = \Sigma_r^{1/2} V_r^T, \tag{4}$$

which satisfies $A^{(0)} B^{(0)} = W'_r$. Starting from this continuous low-rank subspace, we employ ADMM (Boža & Macko, 2025) to alternate between reconstruction-oriented continuous updates and deployable ternary projections. The complete ADMM updates are provided in Appendix A.2.2.

The core projection operator is our Ternary Sign-Value Independent Decomposition (TSVID). Conventional ternary projection typically assumes a single scale $C \approx \alpha T$, which cannot capture heterogeneous row- and column-wise magnitudes. TSVID decomposes a continuous factor as

$$C \approx \mathrm{Diag}(u) T \, \mathrm{Diag}(v), \tag{5}$$

where $T \in \{-1, 0, +1\}^{m \times n}$ captures the ternary topology, while u and v independently model row- and column-wise magnitude variations. This formulation explicitly decouples discrete structure selection from continuous magnitude fitting. For fixed scales, the element-wise ternary update is:

$$T_{ij} = \begin{cases} \mathrm{sign}(C_{ij}), & |C_{ij}| > \frac{1}{2} u_i v_j, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Unlike global thresholding, the decision boundary adapts to the interaction between row and column scales. Alternating updates of T, u, and v yield deterministic ternary factors and continuous scales, with derivations provided in Appendix A.2.3. These factors preserve the low-rank structure inherited from initialization and provide a low-error starting point for subsequent discrete optimization. Stage III then further corrects residual zero/sign assignments through annealed joint refinement.

3.1.3 Stage III: Annealed Joint Ternary Refinement

Stage II provides a deterministic ternary initialization with low reconstruction error, but hard projection may prematurely fix suboptimal zero/sign assignments near decision boundaries. We therefore initialize an annealed joint refinement from the ADMM-TSVID solution, preserving its structural prior while locally revisiting uncertain discrete states. Inspired by the discrete-state decomposition in GSQ (Dadgarnia et al., 2026), each ternary factor is represented as

$$T = M \odot S, \tag{7}$$

where $M \in \{0,1\}^{m\times n}$ controls zero/nonzero states and $S \in \{-1,+1\}^{m\times n}$ determines the signs of activated entries. Given the Stage II initialization $T^{(0)}$, we obtain $M^{(0)}$ and $S^{(0)}$ and initialize their learnable logits around this solution:

$$\theta_{ij}^{(M)} = \lambda_M(2M_{ij}^{(0)} - 1) + \sigma_{\text{init}}\epsilon_{ij}^{(M)}, \qquad \theta_{ij}^{(S)} = \lambda_S M_{ij}^{(0)} S_{ij}^{(0)} + \sigma_{\text{init}}\epsilon_{ij}^{(S)}. \tag{8}$$

At refinement step t, the soft ternary representation is constructed as

$$\widetilde{T}^{(t)} = \widetilde{M}^{(t)} \odot \widetilde{S}^{(t)}, \tag{9}$$

where the relaxed mask and sign are defined as

$$\widetilde{M}^{(t)} = \sigma\left(\frac{\kappa_t\theta^{(M)} + \delta_t\epsilon^{(M)}}{\tau_t}\right), \qquad \widetilde{S}^{(t)} = \tanh\left(\frac{\kappa_t\theta^{(S)} + \delta_t\epsilon^{(S)}}{\tau_t}\right). \tag{10}$$

The relaxation is applied to both factors, yielding $\widetilde{T}_A$ and $\widetilde{T}_B$. During refinement, τ_t decreases from $\tau_{\max}$ to $\tau_{\min}$, κ_t increases from $\kappa_{\min}$ to $\kappa_{\max}$, and δ_t decays to zero, gradually transforming the soft relaxation into discrete ternary assignments. The two factors are jointly optimized under

$$\mathcal{L}_{\text{rec}} = \left\| W' - \text{Diag}(a)\widetilde{T}_A \,\text{Diag}(m)\widetilde{T}_B \,\text{Diag}(b) \right\|_F^2. \tag{11}$$

Unlike the alternating optimization in Stage II, Stage III updates both ternary factors simultaneously, allowing them to collaboratively correct coupled zero/sign decisions. To balance exploration and discretization, we introduce mask hardening, sign hardening, and initialization anchoring:

$$\mathcal{L}_{\text{anneal}} = \mathcal{L}_{\text{rec}} + \beta_M^{(t)}\mathcal{R}_M + \beta_S^{(t)}\mathcal{R}_S + \beta_{\text{anchor}}^{(t)}\mathcal{R}_{\text{anchor}}, \tag{12}$$

with detailed formulations provided in Appendix A.2.4. Discrete topology optimization is further decoupled from continuous magnitude fitting: the ternary logits refine $\widetilde{T}_A$ and $\widetilde{T}_B$, while a, m, and b are updated through closed-form least-squares solutions (Appendix A.2.5). After refinement, the soft factors are hardened into $T_A, T_B \in \{-1, 0, +1\}$.

3.2 Sensitivity-Reweighted Mixed-Rank Allocation (SRMA)

HATF constructs high-fidelity ternary representations for individual linear layers, but a uniform rank assumes similar structural redundancy and compression sensitivity across Transformer modules. We therefore propose Sensitivity-Reweighted Mixed-Rank Allocation (SRMA), which distributes rank capacity under a global bits-per-weight (BPW) budget by jointly considering rank-component structural irreplaceability, matrix-level task sensitivity, and storage cost. To reduce scale bias across heterogeneous modules, structurally comparable matrices are organized into groups $\mathcal{G}_g = \{u = (\ell, t)\}$, where ℓ and t denote the layer index and matrix type. For each matrix u, HATF is first executed with a high probe rank r_u^{probe} to obtain

$$\widehat{W}'_{u,\text{probe}} = P_u \,\text{Diag}(m_u) R_u = \sum_{k=1}^{r_u^{\text{probe}}} m_{u,k} Z_{u,k}, \tag{13}$$

where $P_u = \mathrm{Diag}(a_u)T_{A,u}$, $R_u = T_{B,u}\,\mathrm{Diag}(b_u)$, and $Z_{u,k} = P_{u,:,k}R_{u,k,:}$ is the k-th rank-one basis. Ranking components only by magnitude or isolated energy ignores correlations among bases, since the contribution of a removed component may be compensated by the remaining ones after refitting. We therefore define a refit-aware deletion loss. For a retained set $\mathcal{S}$, let

$$D_u(\mathcal{S}) = \min_{m_{\mathcal{S}}} \left\| W_u'^{,\mathrm{tar}} - \sum_{j\in\mathcal{S}} m_j Z_{u,j} \right\|_F^2, \qquad \Delta_{u,k}^{\mathrm{refit}} = D_u(\mathcal{S}_u \setminus \{k\}) - D_u(\mathcal{S}_u), \tag{14}$$

where $W_u'^{,\mathrm{tar}} = \widehat{W}'_{u,\mathrm{probe}}$. A larger $\Delta_{u,k}^{\mathrm{refit}}$ indicates that component k is less replaceable by the remaining bases. The deletion losses are evaluated efficiently in the compact Gram space, as detailed in Appendix A.2.6. Because an identical reconstruction perturbation can have different task-level effects across matrices, we further estimate matrix sensitivity using a calibration-only scalar gate. For $Y_u = g_u f_u(X_u)$ with $g_u = 1$, we compute

$$G_u = \frac{1}{|\mathcal{B}|}\sum_{b\in\mathcal{B}} \left(\frac{\partial \mathcal{L}_b}{\partial g_u}\right)^2, \qquad \omega_u = \frac{G_u}{\frac{1}{|\mathcal{G}_g|}\sum_{v\in\mathcal{G}_g} G_v}. \tag{15}$$

The group-wise normalization makes ω_u a relative sensitivity measure and reduces gradient-scale differences across matrix types. We combine structural irreplaceability and task sensitivity as

$$I_{u,k} = \omega_u^{\gamma}\Delta_{u,k}^{\mathrm{refit}}, \tag{16}$$

where $\gamma \geq 0$ controls the influence of task sensitivity. This balances structural importance and task sensitivity. Finally, because rank increments incur different storage costs for matrices of different shapes, SRMA allocates rank in blocks according to the utility density

$$\rho_{u,\mathcal{C}} = \frac{\sum_{k\in\mathcal{C}} I_{u,k}}{|\mathcal{C}|\,(b_T(d_{\mathrm{out},u} + d_{\mathrm{in},u}) + b_S)}. \tag{17}$$

The denominator measures the deployment storage introduced by candidate block $\mathcal{C}$, including ternary factor entries and latent scaling coefficients, where b_T and b_S denote the storage cost per entry. The global BPW target is converted into group budgets according to parameter counts. Starting from a rank floor for each matrix, SRMA greedily allocates feasible blocks with the highest $\rho_{u,\mathcal{C}}$ until the budget is exhausted. The resulting mixed-rank model concentrates capacity on structurally irreplaceable components in task-sensitive matrices while accounting for storage efficiency.

3.3 Task-Aligned Ternary Refinement (TATR)

After HATF and SRMA, the target linear layers have compact ternary factors with adaptive ranks. However, weight-space reconstruction remains only a surrogate for the final language-modeling objective, and local errors can accumulate through deep Transformer layers. We therefore propose Task-Aligned Ternary Refinement (TATR), which aligns the compressed model with the full-precision model while preserving the mixed-rank structure determined by HATF and SRMA.

TATR first performs global scale recalibration with all ternary factors and ranks fixed. Specifically, the continuous parameters $\Theta_{\mathrm{scale}} = \{a_\ell, m_\ell, b_\ell\}_{\ell=1}^{L}$ are jointly optimized under a full-model objective, while the embedding and LM head remain frozen. Unlike the local scale fitting in HATF, this task-aware calibration coordinates effective magnitudes across layers, channels, and rank components without restoring dense full-precision weights. Given a calibration sequence X, let f_T and f_S denote the full-precision teacher and compressed student, with predictive distributions p_T and p_S. We use the token-level distillation objective

$$\mathcal{L}_{\mathrm{KD}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{v\in\mathcal{V}} p_T^{(i,v)} \log p_S^{(i,v)}. \tag{18}$$

Since the teacher is fixed, this objective is equivalent to minimizing $D_{\mathrm{KL}}(p_T \| p_S)$ up to a student-independent constant, directly aligning model-level predictive behavior. Global scale recalibration cannot correct residual errors in hard ternary topology. TATR therefore introduces sparse, constrained topology refinement for ternary factors while preserving mixed-rank structure. For $T \in \{-1, 0, +1\}^{m\times n}$, we maintain a continuous proxy $\widetilde{T}$ and use the straight-through estimator

$$T_{\mathrm{STE}} = T + \widetilde{T} - \mathrm{sg}(\widetilde{T}), \tag{19}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient. The forward pass retains the ternary factor, while gradients are propagated to the proxy during optimization. Rather than allowing unrestricted state changes, TATR exploits the ordered state space $-1 \leftrightarrow 0 \leftrightarrow +1$ and restricts each commit to an adjacent transition:

$$d_{ij} = \mathrm{sign}(\widetilde{T}_{ij} - T_{ij}), \qquad T_{ij}^{\mathrm{cand}} = \mathrm{clip}(T_{ij} + d_{ij}, -1, 1). \tag{20}$$

Thus, a single update permits only $-1 \to 0$, $0 \to -1$, $0 \to +1$, or $+1 \to 0$, keeping refinement local to the existing ternary solution. We use the proxy displacement $q_{ij} = |\widetilde{T}_{ij} - T_{ij}|$ as the confidence of a state change and commit only the most confident fraction p_{commit}:

$$T_{ij}^{\mathrm{new}} = \begin{cases} T_{ij}^{\mathrm{cand}}, & q_{ij} \geq Q_{1-p_{\mathrm{commit}}}(q), \\ T_{ij}, & \text{otherwise.} \end{cases} \tag{21}$$

After each commit, the proxy is reset to the hard state, enabling task-guided modifications rather than unconstrained retraining. To limit memory overhead, proxies are activated only for a subset of linear layers, while other layers continue scale-only calibration (details are provided in Appendix A.2.7). Throughout TATR, the SRMA rank configuration remains fixed. All temporary proxies are discarded after refinement, leaving only hard ternary factors and continuous scales for deployment.

4 Experiments

4.1 Experiment setup

Models and Datasets. We evaluate Ternova on representative LLM families, including LLaMA-3 (Grattafiori et al., 2024), Qwen3/3.5/3.8 (Yang et al., 2025; Qwen Team, 2026a;b), and Gemma-3/4 (Team et al., 2025; 2026). We report WikiText2 perplexity (Merity et al., 2016) and seven zero-shot tasks: ARC-Challenge/Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), LAMBADA-openai/standard (Paperno et al., 2016), PIQA (Bisk et al., 2019), and WinoGrande (Sakaguchi et al., 2019). We further evaluate Ternova on more challenging reasoning benchmarks, including the multi-domain knowledge task MMLU (Hendrycks et al., 2021), the mathematical reasoning benchmark GSM8K (Cobbe et al., 2021), and the code generation benchmark HumanEval (Chen et al., 2021).

Baseline Methods. We compare Ternova against representative PTQ methods spanning conventional low-bit quantization and open-source binary/ternary PTQ methods. Specifically, GPTQ (Frantar et al., 2022) is included as a widely adopted integer PTQ baseline, while BiLLM (Huang et al., 2024), STBLLM (Dong et al., 2024), ARB-LLM (Li et al., 2024), HBLLM (Chen et al., 2025), and TWLA (Zhao et al., 2026) represent competitive binary/ternary PTQ methods (with their effective BPW calculated in Appendix A.3). To provide additional reference points in a comparable low-bit regime, we further include recent QAT methods, including OneBit (Xu et al., 2024) and LittleBit (Lee et al., 2026). Since QAT requires additional training data and substantially higher optimization cost, these results are included as reference points rather than strict PTQ comparisons.

Implementation Details. Experiments are implemented with PyTorch and conducted on NVIDIA RTX PRO 6000 GPUs. Following the calibration setting of DBF (Boža & Macko, 2025), we randomly sample 256 sequences from the RedPajama (Weber et al., 2024) dataset with a sequence length of 2048. We quantize only the linear layers in Transformer blocks, while keeping the embedding layer and LM head in full precision. All evaluations use lm-evaluation-harness. See Appendix A.4.1 for hyperparameters and Appendix A.4.3 for calibration-data ablations.

4.2 Main Results

Comparison Results. We evaluate Ternova across LLaMA, Qwen, and Gemma models spanning 4B–70B parameters. As shown in Table 1, despite requiring fewer effective bits per weight than existing low-bit PTQ methods (2.51–3.50 BPW), Ternova preserves near-BF16 performance and achieves the best or second-best results across almost all settings. Here, BPW denotes effective deployment storage cost, including packed ternary factors and retained scales. On LLaMA-3-8B, Ternova achieves 60.97% average zero-shot accuracy at 1.59 BPW, improving over TWLA (2.92 BPW) by 2.72 percentage points while maintaining WikiText2 PPL of 10.94. On LLaMA-3-70B, Ternova reaches 68.23% average accuracy with PPL of 7.49. On Qwen3.5-27B, it achieves 74.62%

Table 1: Comparison of perplexity on WikiText2 and averaged accuracy on seven zero-shot tasks. Bold and underline denote the **best** and second-best quantized results.

Method	BPW	LLaMA-3-8B		LLaMA-3-70B		Gemma-3-4B-it		Gemma-3-12B-it		Qwen3-8B		Qwen3-14B		Qwen3.5-9B		Qwen3.5-27B	
		0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)	0-shot[7] Avg.(↑)	Wiki (↓)
BF16	16	72.51	6.14	79.08	2.86	67.12	17.39	72.20	25.20	69.09	9.00	72.47	8.64	70.67	8.65	75.97	6.90
GPTQ	2.28	26.38	154.38	45.59	24.53	28.07	584.39	26.72	8.21e5	25.80	152.25	28.77	47.48	27.95	71.01	47.02	18.15
BiLLM	2.88	35.38	48.18	28.32	137.64	32.33	134.42	38.41	381.19	33.70	96.21	53.53	28.66	49.99	24.59	67.96	12.21
STBLLM$_{(4:8)}$	3.50	29.43	196.03	26.55	963.92	29.31	202.54	31.46	4.69e5	28.02	291.76	40.88	48.80	42.47	35.85	64.74	13.96
ARB-LLM$_{RC}$	2.51	46.21	28.97	60.76	11.10	41.83	95.18	47.15	4.68e3	49.85	40.90	61.60	18.40	55.02	21.29	70.99	11.55
HBLLM$_{row}$	3.25	55.35	14.26	53.19	9.55	50.35	43.05	57.40	758.45	54.76	16.28	64.12	11.01	64.60	10.99	73.14	8.12
TWLA	2.92	58.25	**10.21**	67.95	**6.88**	50.79	38.27	59.33	482.10	**61.78**	**12.63**	67.35	10.48	63.27	11.52	72.94	8.63
Ours	**1.59**	**60.97**	10.94	**68.23**	7.49	**51.61**	**20.88**	**60.99**	**20.25**	61.41	12.93	**67.80**	**10.05**	**66.15**	**10.17**	**74.62**	**7.30**

average accuracy at 1.59 BPW, approaching BF16 baseline (75.97%) with lower weight storage. These results demonstrate that Ternova preserves LLM capabilities under extreme compression with favorable accuracy–compression trade-off across architectures. Full results and storage-matched comparison with DBF+PV are provided in Appendices A.4.2 and A.4.4, respectively.

Evaluation on Complex Reasoning Tasks. Quantization often introduces more severe degradation on complex reasoning, mathematical problem solving, and code generation tasks. To further evaluate whether Ternova preserves these advanced capabilities under extreme compression, we conduct experiments on Gemma-4-31B-it and Qwen3.8-27B using MMLU, HumanEval, and GSM8K. As shown in Table 2, Ternova consistently achieves the best performance among all quantized methods while substantially reducing model storage. On Qwen3.8-27B, Ternova compresses the BF16 model from 55.59GB to only 10.10GB, while achieving 78.50% MMLU, 49.36% HumanEval, and 67.70% GSM8K. Overall, Ternova preserves over 84% of the averaged BF16 performance across these tasks. Similar effectiveness is also observed on Gemma-4-31B-it. These results demonstrate Ternova effectively preserves complex reasoning capabilities under aggressive compression, achieving a favorable accuracy–memory trade-off for practical deployment. Appendix A.5 provides complementary qualitative examples.

Table 2: Results on challenging reasoning benchmarks.

Model	Method	Model Size	MMLU	HumanEval	GSM8K
Gemma-4-31B-it	BF16	62.58GB	79.54	74.09	72.55
	BiLLM	13.58GB	55.40	0	23.49
	STBLLM$_{(4:8)}$	15.89GB	38.21	0	15.76
	ARB-LLM$_{RC}$	12.19GB	62.59	6.08	49.33
	HBLLM$_{row}$	14.96GB	70.12	40.54	64.28
	Ours	**8.76GB**	**73.88**	**47.29**	**67.01**
Qwen3.8-27B	BF16	55.59GB	83.46	78.05	69.90
	BiLLM	14.18GB	59.54	0	26.46
	STBLLM$_{(4:8)}$	16.13GB	40.88	0	18.42
	ARB-LLM$_{RC}$	13.01GB	67.43	9.76	52.01
	HBLLM$_{row}$	15.34GB	75.39	42.68	66.64
	Ours	**10.10GB**	**78.50**	**49.36**	**67.70**

4.3 Ablation Studies

Component Ablation. We analyze the contribution of each component in Ternova using WikiText2 PPL and seven-task average zero-shot accuracy. As shown in Table 3, the vanilla ternary factorization suffers severe degradation, with a PPL of $6.4e2$ and only 26.37 average accuracy. HATF progressively improves the ternary representation quality: Hessian-aware preconditioning reduces the PPL to $1.4e2$, while introducing TSVID-ADMM and annealed refinement further lowers it to 34.47 and 20.88, respectively, with the average accuracy increasing to 55.29. Building on this stronger representation, SRMA improves the limited-budget rank allocation, reducing PPL to 12.75 and raising accuracy to 60.79. Finally, TATR further aligns the compressed model with task-level behavior, achieving the best PPL of 10.17 and average accuracy of 66.15. These results demonstrate that HATF, SRMA, and TATR are complementary, progressively improving ternary representation, capacity allocation, and model-level alignment. Ablations of TATR's scale calibration and topology refinement are provided in Appendix A.4.5.

Table 3: Impact of different components on Qwen3.5-9B.

Variant	HATF			SRMA	TATR	Wiki2	0-shot[7]
	Step1	Step2	Step3			PPL(↓)	Avg.(↑)
Ternary Factorization	✗	✗	✗	✗		6.4e2	26.37
+Hessian-aware	✓	✗	✗			1.4e2	28.79
+TSVID-ADMM	✓	✓	✗			34.47	44.06
+Annealed refinement	✓	✓	✓			20.88	55.29
+SRMA	✓			✓	✗	12.75	60.79
Full Ternova				✓	✓	**10.17**	**66.15**

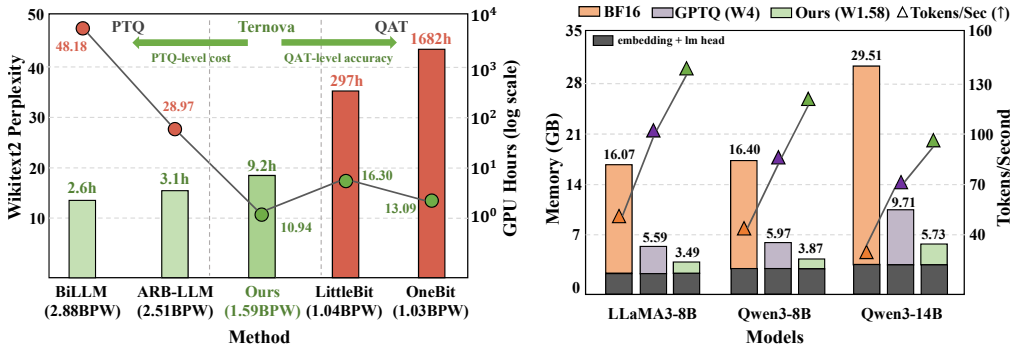


Figure 3: Efficiency Analysis. **Left**: Comparison of quantization time and WikiText2 PPL among representative PTQ/QAT methods and Ternova on LLaMA3-8B. **Right**: Throughput (Tokens/Sec) and memory consumption of BF16, GPTQ (W4), and Ternova (W1.58) across three models.

Rank Allocation Strategy. We compare rank allocation strategies with HATF and TATR fixed under the same BPW budget. As shown in Table 4, both task-sensitivity and refit-aware allocation outperform uniform rank assignment, indicating the importance of non-uniform capacity allocation. Refit-aware allocation reduces WikiText2 PPL from 14.85 to 13.59 and improves the zero-shot average from 58.75 to 59.21 compared with task-sensitivity alone. Combining structural irreplaceability, task sensitivity, and storage cost, SRMA achieves the best performance, reducing PPL to 10.17 and improving average accuracy to 66.15. Group-level rank statistics are reported in Appendix A.4.6.

Table 4: Rank allocation on Qwen3.5-9B.

Allocation Strategy	Wiki2 PPL(↓)	0-shot[7] Avg.(↑)
Uniform Rank	17.42	56.88
Task-Sensitivity	14.85	58.75
Refit-Aware	13.59	59.21
SRMA	**10.17**	**66.15**

4.4 Efficiency Analysis of Ternova

Quantization Time Comparison. We compare Ternova with low-bit PTQ and QAT methods on LLaMA3-8B. In Fig. 3 (left), BiLLM and ARB-LLM require 2.6 and 3.1 GPU hours, respectively, but have higher BPWs and perplexity. Ternova achieves 10.94 WikiText2 PPL at 1.59 BPW in 9.2 GPU hours, outperforming BiLLM (48.18 PPL, 2.88 BPW) and ARB-LLM (28.97 PPL, 2.51 BPW) while retaining PTQ-level efficiency. QAT methods LittleBit and OneBit require 297 and 1682 GPU hours. Despite slightly higher BPW, Ternova achieves lower perplexity in approximately 32× and 183× less time, respectively. Ternova bridges the efficiency–accuracy gap between PTQ and QAT.

Speedup and Memory Savings. We evaluate Ternova with `llama.cpp`[1] on NVIDIA RTX PRO 6000 GPUs. As shown in Fig. 3 (right), Ternova consistently outperforms BF16 and GPTQ (W4) in inference throughput across LLaMA3-8B, Qwen3-8B, and Qwen3-14B, achieving roughly 3× speedup over BF16 and 1.3–1.4× over GPTQ. Ternova also substantially reduces model memory across all evaluated models, achieving up to 80.6% and 41.0% memory savings over BF16 and GPTQ, respectively, while retaining the embedding and LM head in BF16. These results demonstrate substantial practical gains in both inference speed and memory efficiency.

5 Conclusion

We present Ternova, an adaptive low-rank PTQ framework for accurate and efficient ternarization of LLMs. Ternova addresses limited ternary representation capacity, heterogeneous structural sensitivity, and misalignment between local reconstruction and model-level behavior. HATF constructs high-fidelity ternary low-rank representations, SRMA allocates ranks under a fixed BPW budget, and TATR aligns the compressed model with task-level objectives. Experiments across LLM families show that Ternova consistently outperforms low-bit PTQ methods at 1.59 BPW, preserves strong reasoning capabilities, and achieves competitive accuracy with substantially lower quantization cost than QAT-based methods. Ternova delivers practical memory savings and inference acceleration, providing a scalable and deployment-efficient pathway toward ternary LLM compression.

[1] `https://github.com/ggml-org/llama.cpp`

AI USE STATEMENT

In preparing this manuscript, we used large language models (LLMs) as writing assistants for grammar correction, wording refinement, translation, and stylistic polishing. LLMs also assisted in rephrasing and restructuring author-written drafts to improve clarity and readability. The conceptual contributions, technical methods, experimental designs, and analyses remain the responsibility of the authors. All AI-assisted revisions were reviewed by the authors to ensure that they preserved the intended technical meaning. The authors take full responsibility for the accuracy and integrity of the final manuscript, including all AI-assisted content.

ETHICS STATEMENT

This work presents Ternova, a post-training ternarization framework intended to reduce the storage and inference costs of large language models. Experiments use existing pretrained models and calibration and evaluation datasets. While more efficient deployment can improve accessibility, it may also lower barriers to misuse. Compressed models may retain biases, factual errors, or harmful behaviors of their source models, and quantization should not be assumed to preserve existing safety safeguards. Our evaluations focus on task performance and efficiency and do not establish safety, fairness, or privacy guarantees. We therefore encourage application-specific safety and bias assessments before deployment, appropriate safeguards against misuse, and responsible use in accordance with the licenses and usage restrictions of the underlying models and datasets.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we describe the Ternova framework in Sec. 3 and provide detailed pseudocode and derivations in Appendices A.1 and A.2. Sec. 4 specifies the evaluated models, datasets, calibration procedure, quantization scope, evaluation setup, and hardware. Hyperparameter settings for HATF, SRMA, and TATR are reported in Appendix A.4.1. Appendix A.3 details the effective-BPW calculations, including the storage of quantized weights and auxiliary parameters. Detailed task-level results and calibration-data ablations are provided in Appendices A.4.2 and A.4.3, respectively. We also plan to open-source the Ternova implementation, including quantization and evaluation scripts and experimental configurations, after the review process to facilitate reproducibility and future research.

REFERENCES

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

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

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

Vladimír Boža and Vladimír Macko. Two sparse matrices are better than one: Sparsifying neural networks with double sparse factorization, 2025. URL https://arxiv.org/abs/2409.18850.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec

Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

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

Tianqi Chen, Zhe Li, Weixiang Xu, Zeyu Zhu, Dong Li, Lu Tian, Emad Barsoum, Peisong Wang, and Jian Cheng. Ternaryllm: Ternarized large language model. *arXiv preprint arXiv:2406.07177*, 2024.

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, 2018. URL https://arxiv.org/abs/1803.05457.

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

Alireza Dadgarnia, Soroush Tabesh, Mahdi Nikdan, Michael Helcig, Eldar Kurtic, Maximilian Kleinegger, and Dan Alistarh. Gsq: Highly-accurate low-precision scalar quantization for llms via gumbel-softmax sampling, 2026. URL https://arxiv.org/abs/2604.18556.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

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

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.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj,

Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. Billm: Pushing the limit of post-training quantization for llms, 2024. URL https://arxiv.org/abs/2402.04291.

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

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models, 2024. URL https://arxiv.org/abs/2306.02272.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks, 2022. URL https://arxiv.org/abs/1605.04711.

Zhiteng Li, Xianglong Yan, Tianao Zhang, Haotong Qin, Dong Xie, Jiang Tian, zhongchao shi, Linghe Kong, Yulun Zhang, and Xiaokang Yang. Arb-llm: Alternating refined binarizations for large language models, 2024. URL https://arxiv.org/abs/2410.03129.

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.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations, 2025. URL https://arxiv.org/abs/2405.16406.

Xudong Lu, Aojun Zhou, Ziyi Lin, Qi Liu, Yuhui Xu, Renrui Zhang, Xue Yang, Junchi Yan, Peng Gao, and Hongsheng Li. Terdit: Ternary diffusion models with transformers, 2025. URL https://arxiv.org/abs/2405.14854.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

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

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

Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quarterly Journal of Mathematics*, 11(1):50–59, 1960.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context, 2016. URL https://arxiv.org/abs/1606.06031.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026b. URL https://qwen.ai/blog?id=qwen3.8.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. Flatquant: Flatness matters for llm quantization, 2025. URL https://arxiv.org/abs/2410.09426.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh,

Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Biao Zhang, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Jakub Adamek, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Daniele Calandriello, Glenn Cameron, Charlotte Caucheteux, Rahma Chaabouni, Garima Chadha, Jetha Chan, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Sebastian Flennerhag, Takumi Fujimoto, João Gabriel Oliveira, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Tamara von Glehn, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Mayank Gupta, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Ivan Korotkov, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Vivek Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Alon Levkovitch, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Ivan Lobov, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Maciej Mikuła, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Brendan O'Donoghue, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Pan-

igrahy, Unnati Parekh, Nicolas Perez-Nieves, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, George Scrivener, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Bobak Shahriari, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Jean Tarbouriech, Chetan Tekur, Shantanu Thakoor, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Çağlar Ünlü, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Cindy Wu, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Morteza Zadimoghaddam, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks, 2024. URL https://arxiv.org/abs/2402.04396.

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

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2024. URL https://arxiv.org/abs/2211.10438.

He Xiao, Runming Yang, Qingyao Yang, Wendong Xu, Zhen Li, Yupeng Su, Zhengwu Liu, Hongxia Yang, and Ngai Wong. Ptqtp: Post-training quantization to trit-planes for large language models. *arXiv preprint arXiv:2509.16989*, 2025.

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.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. Ternarybert: Distillation-aware ultra-low bit bert, 2020. URL https://arxiv.org/abs/2009.12812.

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

Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained ternary quantization, 2017. URL https://arxiv.org/abs/1612.01064.

A APPENDIX

APPENDIX OVERVIEW

This appendix is organized as follows.

- Section A.1: Pseudocode of Ternova.
  - Section A.1.1: Overall Workflow of Ternova.
  - Section A.1.2: Hessian-Aware Annealed Ternary Factorization (HATF).
  - Section A.1.3: Sensitivity-Reweighted Mixed-Rank Allocation (SRMA).
  - Section A.1.4: Task-Aligned Ternary Refinement (TATR).
- Section A.2: Detailed Derivations and Implementation Details.
  - Section A.2.1: Derivation of Hessian-Aware Preconditioning.
  - Section A.2.2: ADMM Optimization for TSVID-Guided Ternary Factorization.
  - Section A.2.3: Derivation of Ternary Sign-Value Independent Decomposition.
  - Section A.2.4: Annealed Refinement Objective.
  - Section A.2.5: Closed-form Scale Optimization.
  - Section A.2.6: Efficient Optimization of Sensitivity-Reweighted Mixed-Rank Allocation.
  - Section A.2.7: Implementation Details of Task-Aligned Ternary Refinement.
- Section A.3: Effective Bit-Width of Quantized Linear Layers.
- Section A.4: Additional Experimental Details.
  - Section A.4.1: Experimental Hyperparameter Settings.
  - Section A.4.2: Detailed Main Results.
  - Section A.4.3: Ablation on Calibration Data.
  - Section A.4.4: Comparison with DBF.
  - Section A.4.5: Detailed Ablation of Task-Aligned Ternary Refinement.
  - Section A.4.6: Analysis of Mixed-Rank Allocation.
- Section A.5: Qualitative Generation Examples.

A.1 PSEUDOCODE OF TERNOVA

This section provides the detailed pseudocode of Ternova. The complete compression procedure consists of three stages: Hessian-Aware Annealed Ternary Factorization (HATF), Sensitivity-Reweighted Mixed-Rank Allocation (SRMA), and Task-Aligned Ternary Refinement (TATR). HATF constructs high-fidelity ternary low-rank representations for individual linear matrices, SRMA determines adaptive rank configurations under a global storage budget, and TATR further aligns the compressed model with the full-precision model behavior through task-aware refinement.

A.1.1 Overall Workflow of Ternova

Algorithm 1: Ternova: Overall Compression Pipeline

Input: Pretrained LLM $\mathcal{M}$ with L quantized linear matrices, calibration dataset $\mathcal{D}$, target ternary-factor BPW budget $\mathcal{B}$

Output: Compressed ternary LLM $\hat{\mathcal{M}}$

for *each quantized linear matrix* $l = 1, \ldots, L$ **do**

`// Stage 1: Hessian-Aware Annealed Ternary Factorization (HATF)`

Collect calibration activations X_l and output gradients G_l;

Construct the Hessian-aware surrogate weight:

$$W_l' = D_{o,l} W_l D_{i,l}$$

Apply probe-rank TSVID-guided ADMM factorization and annealed refinement to obtain

$$\hat{W}_l' = \mathrm{Diag}(a_l') T_{A,l} \,\mathrm{Diag}(m_l) T_{B,l} \,\mathrm{Diag}(b_l')$$

Absorb the inverse preconditioners into the outer scales:

$$a_l = D_{o,l}^{-1} a_l', \qquad b_l = D_{i,l}^{-1} b_l'$$

`// Stage 2: Sensitivity-Reweighted Mixed-Rank Allocation (SRMA)`

Use the probe-rank HATF representations from Stage 1;

Compute rank-component importance using refit-aware deletion loss;

Estimate matrix-level task sensitivity from calibration data;

Combine structural importance and task sensitivity to obtain allocation scores;

Greedily allocate rank blocks according to storage efficiency until the ternary-factor budget $\mathcal{B}$ is satisfied;

Retain and refit the allocated components to obtain the mixed-rank ternary model $\hat{\mathcal{M}}_{\mathrm{SRMA}}$;

`// Stage 3: Task-Aligned Ternary Refinement (TATR)`

Freeze the mixed-rank configuration obtained from SRMA;

Optimize continuous scale parameters with teacher–student distillation;

Perform sparse topology refinement on selected ternary factors;

Remove all temporary refinement variables;

Assemble the final deployable model:

$$\hat{\mathcal{M}} = \{\mathrm{Diag}(a_l) T_{A,l} \,\mathrm{Diag}(m_l) T_{B,l} \,\mathrm{Diag}(b_l)\}_{l=1}^{L}$$

return $\hat{\mathcal{M}}$

Algorithm 1 summarizes the overall workflow of Ternova. The framework first constructs sensitivity-aware ternary low-rank representations through HATF, then reallocates ternary rank capacity across matrices with SRMA, and finally aligns the compressed model with full-precision behavior through TATR.

A.1.2 Hessian-Aware Annealed Ternary Factorization (HATF)

HATF aims to obtain a high-fidelity ternary low-rank representation for each linear layer. It first embeds task-dependent sensitivity into a surrogate weight space, then performs TSVID-guided ADMM optimization to obtain a reliable ternary initialization, and finally refines ternary topology through annealed joint optimization.

Algorithm 2: Hessian-Aware Annealed Ternary Factorization (HATF)

Input: Full-precision weight matrix W, calibration activations X, output gradients G, target rank r

Output: Ternary factorized representation $\hat{W} = \mathrm{Diag}(a)T_A\,\mathrm{Diag}(m)T_B\,\mathrm{Diag}(b)$

`// Stage I: Hessian-aware preconditioning`

Compute input/output sensitivity statistics:

$$i = \sqrt{\mathbb{E}[X^2] + \epsilon}, \qquad o = \sqrt{\mathbb{E}[G^2] + \epsilon}$$

Construct $D_i = \mathrm{Diag}(i)$, $D_o = \mathrm{Diag}(o)$ and

$$W' = D_o W D_i$$

`// Stage II: TSVID-guided ADMM ternary factorization`

Compute a rank-r SVD initialization:

$$W'_r = U_r \Sigma_r V_r^T$$

Initialize continuous factors:

$$A^0 = U_r \Sigma_r^{1/2}, \qquad B^0 = \Sigma_r^{1/2} V_r^T$$

Apply TSVID to A^0 and B^0 separately and merge their latent-side scales into the middle scale vector m^0;

for $k = 1$ **to** K **do**

- Update continuous factors by reconstruction minimization;
- Update the scaled ternary factors using TSVID projection;
- Update ADMM dual variables;

`// Stage III: Annealed joint ternary refinement`

Initialize ternary logits from the ADMM solution;

for $t = 1$ **to** T **do**

- Generate soft ternary factors;
- Optimize the annealing objective with reconstruction and regularization terms;
- Refit scale parameters and update the annealing schedule;

Convert the refined factors into hard ternary matrices:

$$T_A, T_B \in \{-1, 0, +1\}$$

Let the refined surrogate-space representation be

$$\hat{W}' = \mathrm{Diag}(a')T_A\,\mathrm{Diag}(m)T_B\,\mathrm{Diag}(b')$$

Recover the original-space representation by absorbing inverse preconditioning into the outer scales:

$$a = D_o^{-1}a', \qquad b = D_i^{-1}b', \qquad \hat{W} = D_o^{-1}\hat{W}'D_i^{-1}$$

return $\hat{W}$

Algorithm 2 describes the complete HATF procedure. The Hessian-aware transformation introduces task sensitivity before ternary optimization, while TSVID-guided ADMM provides a stable discrete initialization. The annealed refinement stage further improves discrete topology without introducing additional full-precision parameters.

A.1.3 Sensitivity-Reweighted Mixed-Rank Allocation (SRMA)

SRMA determines the rank configuration of ternary factors under a fixed BPW budget. Instead of assigning identical ranks to all layers, SRMA evaluates the structural importance of each rank component and combines it with matrix-level task sensitivity to allocate ternary capacity efficiently.

Algorithm 3: Sensitivity-Reweighted Mixed-Rank Allocation (SRMA)

Input: Probe-rank ternary representations, calibration dataset $\mathcal{D}$, target ternary-factor BPW budget $\mathcal{B}$

Output: Mixed-rank configuration $\{r_u\}$

for *each quantized linear matrix* u **do**

Generate the probe-rank HATF representation:

$$\hat{W}_u = \sum_{k=1}^{r_u^{\text{probe}}} m_{u,k} Z_{u,k}$$

Evaluate component importance using the refit-aware deletion loss:

$$\Delta_{u,k}^{\text{refit}}$$

Compute the Fisher-style matrix sensitivity:

$$G_u = \frac{1}{|\mathcal{D}|} \sum_{b \in \mathcal{D}} \left(\frac{\partial \mathcal{L}_b}{\partial g_u} \right)^2$$

Normalize G_u within each structurally comparable matrix group to obtain ω_u, and compute

$$I_{u,k} = \omega_u^{\gamma} \Delta_{u,k}^{\text{refit}}$$

Initialize each matrix with its minimum feasible rank allocation;

while *a feasible rank block remains under the budget* **do**

For every feasible candidate block $\mathcal{C}$ of matrix u, compute

$$\rho_{u,\mathcal{C}} = \frac{\sum_{k \in \mathcal{C}} I_{u,k}}{C_{u,\mathcal{C}}}$$

Select the feasible block with the highest allocation density;

Increase the corresponding rank allocation;

Update the remaining ternary-factor budget;

return $\{r_u\}$

Algorithm 3 allocates ternary capacity according to both structural irreplaceability and task relevance. The resulting mixed-rank configuration preserves more capacity for important components while avoiding unnecessary storage overhead.

A.1.4 Task-Aligned Ternary Refinement (TATR)

TATR performs the final model-level calibration after HATF and SRMA. It first optimizes continuous scale parameters with the full-model objective, and then performs sparse ternary topology refinement while keeping the mixed-rank structure unchanged.

Algorithm 4: Task-Aligned Ternary Refinement (TATR)

Input: Mixed-rank ternary model $\hat{\mathcal{M}}$, teacher model $\mathcal{M}$, calibration dataset $\mathcal{D}$
Output: Final ternary model $\hat{\mathcal{M}}$

```
// Stage I: Task-aware scale calibration
```

Freeze ternary topology:

$$T_A, T_B$$

Optimize scale parameters:

$$\Theta_{scale} = \{a_l, m_l, b_l\}$$

using teacher-student distillation loss;

```
// Stage II: Sparse topology refinement
```

Select sensitive ternary factors;
Initialize continuous ternary proxies;
for $t = 1$ **to** T **do**
    Compute task-level loss;
    Update ternary proxies using STE gradients;
    Generate adjacent ternary transitions:

$$-1 \leftrightarrow 0 \leftrightarrow +1$$

    Commit high-confidence topology changes;
    Reset proxies to updated ternary states;
Remove refinement proxies;
return $\hat{\mathcal{M}}$

Algorithm 4 performs final task alignment. Continuous scale calibration corrects magnitude mismatch introduced by compression, while sparse topology refinement modifies only highly confident ternary states. After refinement, only deployable ternary factors and compact scale parameters are retained.

A.2 Detailed Derivations and Implementation Details

A.2.1 Derivation of Hessian-Aware Preconditioning

We provide the detailed derivation of the Hessian-aware preconditioning introduced in Sec. 3.1.1. For a linear layer with weight matrix W, let $\Delta W = W - \widehat{W}$ denote the reconstruction error introduced by compression. Under the standard second-order PTQ approximation, we neglect the first-order term and approximate the loss perturbation as

$$\Delta\mathcal{L} \approx \frac{1}{2}\,\mathrm{vec}(\Delta W)^T H_W\,\mathrm{vec}(\Delta W), \tag{22}$$

where H_W denotes the Hessian with respect to the weight parameters.

Following the Kronecker-factored approximate-curvature approximation (K-FAC) (Martens & Grosse, 2015), the curvature of a linear layer can be approximated by the Kronecker product of the input-activation and output-gradient second moments:

$$H_W \approx \mathbb{E}[xx^T] \otimes \mathbb{E}[gg^T]. \tag{23}$$

Let $A = \mathbb{E}[xx^T]$ and $B = \mathbb{E}[gg^T]$. Using the standard vectorization identity,

$$\mathrm{vec}(\Delta W)^T (A \otimes B)\,\mathrm{vec}(\Delta W) = \left\| B^{1/2} \Delta W A^{1/2} \right\|_F^2. \tag{24}$$

Therefore,

$$\Delta\mathcal{L} \approx \frac{1}{2} \left\| \mathbb{E}[gg^T]^{1/2} (W - \widehat{W}) \mathbb{E}[xx^T]^{1/2} \right\|_F^2. \tag{25}$$

Maintaining the full covariance factors is expensive. We therefore retain only their diagonal entries and define

$$D_i = \mathrm{Diag}(i), \qquad D_o = \mathrm{Diag}(o), \tag{26}$$

with

$$i_j = \sqrt{\mathbb{E}[x_j^2] + \epsilon}, \qquad o_u = \sqrt{\mathbb{E}[g_u^2] + \epsilon}, \tag{27}$$

where a small $\epsilon > 0$ prevents numerical instability when the preconditioners are inverted. Under this diagonal curvature approximation,

$$\Delta\mathcal{L} \approx \frac{1}{2} \left\| D_o(W - \widehat{W}) D_i \right\|_F^2 . \tag{28}$$

Defining the surrogate matrices

$$W' = D_o W D_i, \qquad \widehat{W}' = D_o \widehat{W} D_i, \tag{29}$$

gives

$$D_o(W - \widehat{W}) D_i = W' - \widehat{W}'. \tag{30}$$

Since the constant factor $1/2$ does not affect the minimizer, optimization under the diagonal curvature approximation is equivalent to

$$\min_{\widehat{W}'} \|W' - \widehat{W}'\|_F^2. \tag{31}$$

Thus, the surrogate-space reparameterization is exactly equivalent to the adopted diagonal second-order objective, while the diagonalization itself remains an approximation to the full K-FAC curvature. Subsequent low-rank and ternary optimization can therefore be carried out in a standard Frobenius geometry that preserves the selected input- and output-side sensitivity information.

A.2.2 ADMM Optimization for TSVID-Guided Ternary Factorization

We provide the detailed ADMM optimization used in the TSVID-guided ternary factorization introduced in Sec. 3.1.2. Given the surrogate weight matrix W', ADMM bridges a continuous low-rank reconstruction with deployable scaled ternary factors. We write

$$\min_{A_q, B_q} \frac{1}{2} \|W' - A_q B_q\|_F^2, \tag{32}$$

where A_q and B_q are constrained to the scaled ternary forms produced by TSVID.

To decouple the continuous reconstruction variables from the discrete constraints, we introduce continuous auxiliary variables $\bar{A}, \bar{B}$ and impose

$$\bar{A} = A_q, \qquad \bar{B} = B_q. \tag{33}$$

Using scaled dual variables U_A and U_B, the augmented Lagrangian, up to constants independent of the primal variables, is

$$\begin{aligned} \mathcal{L}_\rho = & \frac{1}{2} \|W' - \bar{A}\bar{B}\|_F^2 \\ & + \frac{\rho}{2} \|\bar{A} - A_q + U_A\|_F^2 + \frac{\rho}{2} \|\bar{B} - B_q + U_B\|_F^2. \end{aligned} \tag{34}$$

The continuous factors are updated by alternating quadratic minimization. With $\bar{B}^{(k)}$ fixed,

$$\bar{A}^{(k+1)} = \left[W'(\bar{B}^{(k)})^T + \rho(A_q^{(k)} - U_A^{(k)}) \right] \left[\bar{B}^{(k)}(\bar{B}^{(k)})^T + \rho I \right]^{-1}. \tag{35}$$

With $\bar{A}^{(k+1)}$ fixed,

$$\bar{B}^{(k+1)} = \left[(\bar{A}^{(k+1)})^T \bar{A}^{(k+1)} + \rho I \right]^{-1} \left[(\bar{A}^{(k+1)})^T W' + \rho(B_q^{(k)} - U_B^{(k)}) \right]. \tag{36}$$

The constrained factors are then updated by TSVID projection:

$$A_q^{(k+1)} = \Pi_{\mathrm{TSVID}}(\bar{A}^{(k+1)} + U_A^{(k)}), \tag{37}$$

$$B_q^{(k+1)} = \Pi_{\mathrm{TSVID}}(\bar{B}^{(k+1)} + U_B^{(k)}). \tag{38}$$

Finally, the scaled dual variables are updated as

$$U_A^{(k+1)} = U_A^{(k)} + \bar{A}^{(k+1)} - A_q^{(k+1)}, \tag{39}$$

$$U_B^{(k+1)} = U_B^{(k)} + \bar{B}^{(k+1)} - B_q^{(k+1)}. \tag{40}$$

Through these alternating continuous updates, TSVID projections, and dual updates, ADMM progressively reduces the mismatch between the continuous low-rank approximation and the deployable scaled ternary representation. After separately projecting the two factors, the two latent-side scale vectors are multiplied element-wise and merged into the single middle scale vector m used by Ternova.

A.2.3 Derivation of Ternary Sign-Value Independent Decomposition

We provide the detailed derivation of TSVID introduced in Sec. 3.1.2. As the projection operator in the TSVID-guided ADMM optimization, TSVID approximates a continuous matrix $C \in \mathbb{R}^{m\times n}$ by row-wise and column-wise scales together with a ternary topology:

$$\min_{u,v,T} \|C - \mathrm{Diag}(u)T\,\mathrm{Diag}(v)\|_F^2, \qquad T \in \{-1, 0, +1\}^{m\times n}. \tag{41}$$

Because signs of the scale vectors can always be absorbed into rows or columns of T, we use the canonical representation $u \geq 0$ and $v \geq 0$ when deriving the ternary decision rule.

For fixed u and v, the optimization over T decomposes into independent element-wise problems. Let

$$s_{ij} = u_i v_j \geq 0. \tag{42}$$

If $T_{ij} = 0$, the reconstruction error at position (i, j) is

$$e_0 = C_{ij}^2. \tag{43}$$

For a non-zero ternary state, the optimal sign is $\mathrm{sign}(C_{ij})$, yielding

$$e_1 = (|C_{ij}| - s_{ij})^2. \tag{44}$$

The non-zero state is preferred when $e_1 < e_0$, which is equivalent to

$$|C_{ij}| > \frac{1}{2} s_{ij}. \tag{45}$$

Therefore,

$$T_{ij} = \begin{cases} \mathrm{sign}(C_{ij}), & |C_{ij}| > \frac{1}{2} u_i v_j, \\ 0, & \text{otherwise.} \end{cases} \tag{46}$$

Unlike a global ternary threshold, TSVID adapts the decision boundary to the interaction between row- and column-wise magnitudes.

With T and one scale vector fixed, the other scale vector admits a least-squares update. For the row-wise scales,

$$u_i = \frac{\sum_j C_{ij} T_{ij} v_j}{\sum_j (T_{ij} v_j)^2 + \epsilon}. \tag{47}$$

Similarly,

$$v_j = \frac{\sum_i C_{ij} u_i T_{ij}}{\sum_i (u_i T_{ij})^2 + \epsilon}. \tag{48}$$

After each least-squares update, any negative scale sign is absorbed into the corresponding row or column of T, restoring the canonical non-negative scale representation. Because the factorization also has a multiplicative gauge ambiguity, we apply gauge normalization to prevent one scale vector from growing while the other shrinks proportionally.

By alternating ternary-state updates and scale refitting, TSVID obtains a stable scaled ternary approximation with element-wise adaptive ternary thresholds.

A.2.4 ANNEALED REFINEMENT OBJECTIVE

We provide the detailed formulation of the annealed refinement objective introduced in Section 3.1.3. Given the soft ternary factors $\widetilde{T}_A$ and $\widetilde{T}_B$, the objective is designed to improve reconstruction fidelity, encourage convergence to discrete ternary states, and preserve the reliable structural prior obtained from Stage II.

For each ternary factor,

$$\widetilde{T}_X = \widetilde{M}_X \odot \widetilde{S}_X, \qquad X \in \{A, B\}, \tag{49}$$

where $\widetilde{M}_X \in [0, 1]$ denotes the soft non-zero mask and $\widetilde{S}_X \in [-1, 1]$ denotes the soft sign state. The Stage-II initialization is written as $T_X^{(0)} = M_X^{(0)} \odot S_X^{(0)}$, with $M_X^{(0)} \in \{0, 1\}$ and $S_X^{(0)} \in \{-1, +1\}$ on active entries.

The mask-hardening regularizer is

$$\mathcal{R}_M = \sum_{X \in \{A,B\}} \sum_{i,j} \widetilde{M}_{X,ij}(1 - \widetilde{M}_{X,ij}), \tag{50}$$

which is minimized as the soft masks approach 0 or 1. The sign-hardening term is

$$\mathcal{R}_S = \sum_{X \in \{A,B\}} \sum_{i,j} \widetilde{M}_{X,ij}(1 - \widetilde{S}_{X,ij}^2), \tag{51}$$

where mask weighting avoids constraining sign variables at inactive positions.

Because Stage III is intended as a local refinement rather than a complete discrete search, we further anchor the soft variables to the Stage-II solution:

$$\begin{aligned} \mathcal{R}_{\text{anchor}} = &\sum_{X \in \{A,B\}} \|\widetilde{M}_X - M_X^{(0)}\|_F^2 \\ &+ \sum_{X \in \{A,B\}} \|M_X^{(0)} \odot (\widetilde{S}_X - S_X^{(0)})\|_F^2. \end{aligned} \tag{52}$$

The first term penalizes large changes to zero/non-zero topology, while the second preserves the signs of initially active entries.

Combining these terms with reconstruction loss yields

$$\mathcal{L}_{\text{anneal}} = \mathcal{L}_{\text{rec}} + \beta_M^{(t)} \mathcal{R}_M + \beta_S^{(t)} \mathcal{R}_S + \beta_{\text{anchor}}^{(t)} \mathcal{R}_{\text{anchor}}. \tag{53}$$

For $t = 1, \ldots, T$, define the normalized progress

$$s_t = \frac{t - 1}{T - 1}. \tag{54}$$

The hardening weights increase linearly,

$$\beta_M^{(t)} = \beta_M^0 + (\beta_M^1 - \beta_M^0) s_t, \tag{55}$$

$$\beta_S^{(t)} = \beta_S^0 + (\beta_S^1 - \beta_S^0) s_t, \tag{56}$$

while the anchoring coefficient decays as

$$\beta_{\text{anchor}}^{(t)} = \beta_{\text{anchor}}^0 (1 - s_t). \tag{57}$$

This schedule keeps early optimization close to the reliable Stage-II structure, then gradually relaxes the anchor while strengthening ternary hardening. Together with the temperature, logit-amplification, and perturbation schedules described in Section 3.1.3, it provides a smooth transition from continuous refinement to discrete ternary deployment.

A.2.5 CLOSED-FORM SCALE OPTIMIZATION

We provide the detailed derivation of the closed-form scale updates used in Stage III. Given the refined soft ternary factors $\widetilde{T}_A$ and $\widetilde{T}_B$, the reconstruction objective is formulated as

$$\min_{a,m,b} \left\| W' - \mathrm{Diag}(a)\widetilde{T}_A \,\mathrm{Diag}(m)\widetilde{T}_B \,\mathrm{Diag}(b) \right\|_F^2 . \tag{58}$$

Since the discrete structures are optimized through ternary logits, the continuous scale parameters are updated separately through closed-form least-squares solutions.

For simplicity, we denote

$$X = \widetilde{T}_A, \qquad Y = \widetilde{T}_B. \tag{59}$$

First, consider the update of the output-side scale vector a. Fixing X, Y, m, and b, the reconstruction can be rewritten as

$$W' \approx \mathrm{Diag}(a)H, \tag{60}$$

where

$$H = X \,\mathrm{Diag}(m)Y \,\mathrm{Diag}(b). \tag{61}$$

For the i-th output channel, the optimization problem becomes

$$a_i = \arg\min_a \|W'_{i,:} - aH_{i,:}\|_2^2. \tag{62}$$

Taking the derivative with respect to a and setting it to zero gives

$$a_i = \frac{\langle W'_{i,:}, H_{i,:} \rangle}{\|H_{i,:}\|_2^2 + \epsilon}. \tag{63}$$

Similarly, for the input-side scale vector b, fixing X, Y, m, and a, we define

$$G = \mathrm{Diag}(a)X \,\mathrm{Diag}(m)Y. \tag{64}$$

The update of each input channel scale is obtained by solving

$$b_j = \arg\min_b \|W'_{:,j} - bG_{:,j}\|_2^2, \tag{65}$$

which gives

$$b_j = \frac{\langle W'_{:,j}, G_{:,j} \rangle}{\|G_{:,j}\|_2^2 + \epsilon}. \tag{66}$$

The middle scale vector m requires a joint update over the latent rank dimension. Let

$$P = \mathrm{Diag}(a)X, \qquad R = Y \,\mathrm{Diag}(b). \tag{67}$$

The reconstruction term involving m becomes

$$P \,\mathrm{Diag}(m)R = \sum_{k=1}^{r} m_k P_{:,k} R_{k,:}. \tag{68}$$

Therefore, updating m is equivalent to solving a linear least-squares problem:

$$\min_{m} \|W' - P\,\mathrm{Diag}(m)R\|_F^2 . \tag{69}$$

Using the vectorization identity, this objective can be rewritten as

$$\min_{m} \|\,\mathrm{vec}(W') - (R^T \odot_{\mathrm{KR}} P)m\|_2^2, \tag{70}$$

where $\odot_{\mathrm{KR}}$ denotes the Khatri–Rao product.

The corresponding normal equation is

$$(K + \epsilon I)m = h, \tag{71}$$

where

$$K = (P^T P) \circ (RR^T), \tag{72}$$

where $\circ$ denotes the Hadamard product, and

$$h = \mathrm{diag}(P^T W' R^T). \tag{73}$$

Therefore, the closed-form solution of the middle scale vector is

$$m = (K + \epsilon I)^{-1} h. \tag{74}$$

The computation only involves an $r \times r$ Gram matrix rather than explicitly constructing all rank-one basis matrices. Since the rank r is substantially smaller than the original dimensions of the weight matrix, the proposed scale update introduces negligible overhead compared with the ternary factor optimization.

By alternating the updates of a, m, and b with the ternary structure refinement, the continuous magnitude parameters are adaptively fitted to the current ternary factors, enabling effective decoupling between discrete topology optimization and continuous amplitude reconstruction.

A.2.6 Efficient Optimization of Sensitivity-Reweighted Mixed-Rank Allocation

We provide additional derivations and implementation details for the refit-aware deletion loss, matrix-level sensitivity estimation, and cost-aware mixed-rank allocation introduced in Section 3.2.

Gram-space refit-aware evaluation. For a matrix u, SRMA starts from the probe-rank HATF representation

$$\widehat{W}'_{u,\mathrm{probe}} = P_u\,\mathrm{Diag}(m_u)R_u = \sum_{k=1}^{r_u^{\mathrm{probe}}} m_{u,k} Z_{u,k}, \tag{75}$$

where $Z_{u,k} = P_{u,:,k} R_{u,k,:}$ is the k-th rank-one basis. During rank evaluation, P_u and R_u are fixed, while the middle-scale coefficients associated with the retained rank components are refitted.

Let

$$W'^{,\mathrm{tar}}_u = \widehat{W}'_{u,\mathrm{probe}} \tag{76}$$

denote the probe-rank HATF representation used as the target matrix for rank evaluation. Since SRMA selects rank components from the candidate basis generated by HATF rather than constructing a new representation, evaluating component deletion within this probe subspace directly measures the marginal contribution of each available rank component after refitting.

For a retained set $\mathcal{S}$, the reconstruction objective is

$$D_u(\mathcal{S}) = \min_{m_\mathcal{S}} \left\| W_u'^{,\mathrm{tar}} - \sum_{j\in\mathcal{S}} m_j Z_{u,j} \right\|_F^2 . \tag{77}$$

Directly solving this problem in the full weight space for every candidate deletion would be expensive. We therefore exploit the low-dimensional Gram structure of the rank-one bases. Define

$$K_u = (P_u^T P_u) \circ (R_u R_u^T), \qquad h_u = \mathrm{diag}\left(P_u^T W_u'^{,\mathrm{tar}} R_u^T\right), \tag{78}$$

where $\circ$ denotes the Hadamard product. Each entry of K_u corresponds to the Frobenius inner product between two rank-one bases:

$$[K_u]_{ij} = \langle Z_{u,i}, Z_{u,j}\rangle_F, \qquad [h_u]_i = \langle W_u'^{,\mathrm{tar}}, Z_{u,i}\rangle_F. \tag{79}$$

Let $K_{u,\mathcal{SS}}$ and $h_{u,\mathcal{S}}$ denote the corresponding submatrix and subvector for the retained set. The optimal middle-scale coefficients satisfy

$$K_{u,\mathcal{SS}} m^*_{u,\mathcal{S}} = h_{u,\mathcal{S}}. \tag{80}$$

In practice, a small damping term λI is added for numerical stability:

$$m^*_{u,\mathcal{S}} = \left(K_{u,\mathcal{SS}} + \lambda I\right)^{-1} h_{u,\mathcal{S}}. \tag{81}$$

The reconstruction distortion can then be evaluated without explicitly reconstructing the full weight matrix:

$$\begin{aligned} D_u(\mathcal{S}) = \ & \left\|W_u'^{,\mathrm{tar}}\right\|_F^2 - 2h_{u,\mathcal{S}}^T m^*_{u,\mathcal{S}} \\ & + (m^*_{u,\mathcal{S}})^T K_{u,\mathcal{SS}} m^*_{u,\mathcal{S}}. \end{aligned} \tag{82}$$

Therefore, the refit-aware deletion loss

$$\Delta^{\mathrm{refit}}_{u,k} = D_u(\mathcal{S}_u \setminus \{k\}) - D_u(\mathcal{S}_u) \tag{83}$$

can be computed entirely in a Gram space of dimension at most $r_u^{\mathrm{probe}} \times r_u^{\mathrm{probe}}$ rather than the original $d_{\mathrm{out},u} \times d_{\mathrm{in},u}$ weight space.

For the undamped least-squares problem, the deletion loss also admits a compact Schur-complement form. Let

$$Q_{u,\mathcal{S}} = K^{-1}_{u,\mathcal{SS}}, \qquad m^*_{u,\mathcal{S}} = Q_{u,\mathcal{S}} h_{u,\mathcal{S}}. \tag{84}$$

Then the increase in optimal reconstruction distortion after deleting component $k \in \mathcal{S}$ is

$$\Delta^{\mathrm{refit}}_{u,k} = \frac{\left(m^*_{u,k}\right)^2}{[Q_{u,\mathcal{S}}]_{kk}}. \tag{85}$$

This form explicitly shows that the importance of a rank component depends not only on its coefficient magnitude but also on its correlation with the remaining bases. In practice, the damped Gram system is used to improve numerical robustness.

Matrix-level task sensitivity. The refit-aware deletion loss measures structural irreplaceability within an individual matrix but does not capture how strongly perturbations of that matrix affect the model-level objective. We therefore introduce a calibration-only scalar gate g_u at the output of each matrix:

$$Y_u = g_u f_u(X_u), \qquad g_u = 1. \tag{86}$$

For calibration sample b, the derivative with respect to this gate can be written as

$$\frac{\partial \mathcal{L}_b}{\partial g_u} = \left\langle \frac{\partial \mathcal{L}_b}{\partial Y_u}, f_u(X_u) \right\rangle. \tag{87}$$

We then estimate the Fisher-style sensitivity of matrix u as

$$G_u = \frac{1}{|\mathcal{D}|} \sum_{b \in \mathcal{D}} \left(\frac{\partial \mathcal{L}_b}{\partial g_u} \right)^2 . \tag{88}$$

Because raw gradient magnitudes can differ systematically across module types, SRMA normalizes the sensitivity within each candidate group:

$$\omega_u = \frac{G_u}{\frac{1}{|\mathcal{G}_g|} \sum_{v \in \mathcal{G}_g} G_v}. \tag{89}$$

For $u \in \mathcal{G}_g$, the resulting ω_u therefore measures the relative task sensitivity of matrix u among structurally comparable and functionally related matrices.

Combining local structural irreplaceability with matrix-level task sensitivity gives the component-level retention importance

$$I_{u,k} = \omega_u^{\gamma} \Delta_{u,k}^{\mathrm{refit}}, \tag{90}$$

where γ controls the contribution of model-level sensitivity.

Cost-aware block allocation. A rank component of matrix u introduces one additional column in $T_{A,u}$ and one additional row in $T_{B,u}$, while also requiring one additional latent scaling coefficient in m_u. Therefore, the marginal storage cost of one rank component is

$$c_u = b_{\mathrm{T}} \left(d_{\mathrm{out},u} + d_{\mathrm{in},u} \right) + b_{\mathrm{S}}, \tag{91}$$

where b_{T} denotes the effective storage cost of each ternary factor entry and b_{S} denotes the storage cost of one latent scaling coefficient. Since the continuous scaling vectors are stored in BF16, we set $b_{\mathrm{S}} = 16$.

For efficient hardware-friendly allocation, rank components are assigned in blocks of size q. After sorting the candidate rank components of each matrix by their retention importance, the utility of a candidate block $\mathcal{C}$ is

$$S_{u,\mathcal{C}} = \sum_{k \in \mathcal{C}} I_{u,k}, \tag{92}$$

while its deployment storage cost is

$$C_{u,\mathcal{C}} = |\mathcal{C}| \left[b_{\mathrm{T}} \left(d_{\mathrm{out},u} + d_{\mathrm{in},u} \right) + b_{\mathrm{S}} \right]. \tag{93}$$

The allocation density is therefore

$$\rho_{u,\mathcal{C}} = \frac{S_{u,\mathcal{C}}}{C_{u,\mathcal{C}}} = \frac{\sum_{k \in \mathcal{C}} I_{u,k}}{|\mathcal{C}| \left[b_{\mathrm{T}} (d_{\mathrm{out},u} + d_{\mathrm{in},u}) + b_{\mathrm{S}} \right]}. \tag{94}$$

Let N_g denote the total number of original weight parameters in candidate group $\mathcal{G}_g$:

$$N_g = \sum_{u \in \mathcal{G}_g} d_{\mathrm{out},u} d_{\mathrm{in},u}. \tag{95}$$

Given a target effective BPW b_{tar}, the corresponding storage budget of group g is

$$B_g = b_{\mathrm{tar}} N_g. \tag{96}$$

Thus, the global BPW target induces proportional group budgets according to the original parameter counts while accounting for both ternary factors and continuous latent scales.

To prevent individual matrices from collapsing to excessively small ranks, each matrix is initialized with a small rank floor. Let

$$r_u^{\mathrm{uni}} = \frac{b_{\mathrm{tar}} d_{\mathrm{out},u} d_{\mathrm{in},u} - b_{\mathrm{S}}(d_{\mathrm{out},u} + d_{\mathrm{in},u})}{b_{\mathrm{T}}(d_{\mathrm{out},u} + d_{\mathrm{in},u}) + b_{\mathrm{S}}} \tag{97}$$

denote the approximate rank corresponding to uniform allocation under the same effective storage budget. The practical rank floor is defined as

$$r_u^{\mathrm{floor}} = q \left\lceil \frac{\max\left(r_{\min}, \lceil \eta r_u^{\mathrm{uni}} \rceil\right)}{q} \right\rceil, \tag{98}$$

where $r_{\min}$ is the minimum allowed rank, η controls the relative floor, and the outer ceiling rounds the result upward to the next feasible rank-block boundary. This upward rounding preserves the intended lower-bound interpretation of the rank floor.

Starting from these rank floors, SRMA repeatedly selects the feasible candidate block with the largest $\rho_{u,\mathcal{C}}$ within each group while respecting the remaining group budget. A prefix constraint is imposed such that a later block of a matrix can only be selected after all preceding blocks of that matrix have been allocated. The procedure terminates when no feasible block can be added without exceeding the corresponding effective storage budget.

This allocation strategy approximates the constrained capacity-allocation problem

$$\max_{\{\mathcal{C}\}} \sum_u \sum_{\mathcal{C}\in\mathcal{A}_u} S_{u,\mathcal{C}} \qquad \text{s.t.} \qquad \sum_u \sum_{\mathcal{C}\in\mathcal{A}_u} C_{u,\mathcal{C}} \le B, \tag{99}$$

where $\mathcal{A}_u$ denotes the selected rank blocks for matrix u and $B = \sum_g B_g$ is the total deployment storage budget. Consequently, SRMA allocates limited ternary capacity according to structural irreplaceability, model-level task sensitivity, and marginal storage efficiency rather than enforcing a uniform rank across the model.

A.2.7 Implementation Details of Task-Aligned Ternary Refinement

This appendix provides the implementation details of the Task-Aligned Ternary Refinement (TATR) introduced in Section 3.3. Unlike conventional post-training optimization that updates dense weight parameters, TATR only optimizes the continuous scale parameters and a small number of temporary ternary proxies while preserving the mixed-rank ternary structure determined by HATF and SRMA.

Scale-only Task Calibration. During the first stage of TATR, all ternary factors remain fixed:

$$T_{A,\ell}, T_{B,\ell} \in \{-1, 0, +1\}. \tag{100}$$

Only the continuous scale parameters are updated:

$$\Theta_{\mathrm{scale}} = \{a_\ell, m_\ell, b_\ell\}_{\ell=1}^{L}. \tag{101}$$

The optimization objective is the token-level knowledge distillation loss. Let N denote the number of calibration-token positions used in the objective and $\mathcal{V}$ the vocabulary:

$$\min_{\Theta_{\mathrm{scale}}} \mathcal{L}_{\mathrm{KD}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{v\in\mathcal{V}} p_T^{(i,v)} \log p_S^{(i,v)}. \tag{102}$$

Since the ternary topology and rank allocation are fixed, the optimization only adjusts the magnitude parameters associated with the existing ternary representation. Therefore, the number of trainable parameters is proportional to the sum of input dimension, output dimension, and allocated rank, rather than the original dense weight size.

Straight-Through Ternary Proxy Optimization. To enable task-driven correction of discrete ternary states, TATR introduces temporary continuous proxies for selected ternary factors. For a ternary matrix T, the proxy $\widetilde{T}$ is initialized as

$$\widetilde{T}^{(0)} = T. \tag{103}$$

During forward propagation, the hard ternary factor is preserved through the straight-through formulation:

$$T_{\mathrm{STE}} = T + \widetilde{T} - \mathrm{sg}(\widetilde{T}), \tag{104}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient. Consequently, the forward value is exactly T, while the backward path supplies an identity surrogate gradient to $\widetilde{T}$.

The proxy is optimized together with the scale parameters using

$$\min_{\Theta_{\mathrm{scale}},\widetilde{T}} \mathcal{L}_{\mathrm{KD}}(f_S(X;\Theta_{\mathrm{scale}},T_{\mathrm{STE}}),f_T(X)). \tag{105}$$

Adjacent-State Topology Update. Directly replacing a ternary state from -1 to $+1$ or vice versa may introduce unstable discrete changes. Therefore, TATR restricts topology updates to adjacent ternary states. For each element, the transition direction is determined by

$$d_{ij} = \mathrm{sign}(\widetilde{T}_{ij} - T_{ij}), \tag{106}$$

and the candidate state is generated as

$$T_{ij}^{\mathrm{cand}} = \mathrm{clip}(T_{ij} + d_{ij}, -1, 1). \tag{107}$$

This formulation allows only

$$-1 \to 0, \quad 0 \to -1, \quad 0 \to +1, \quad +1 \to 0, \tag{108}$$

while preventing direct sign reversal. Therefore, topology refinement remains a local search around the HATF initialization.

Sparse Topology Commit. To avoid excessive structural changes during task optimization, TATR does not update all candidate positions simultaneously. Instead, the confidence of each possible state transition is measured by

$$q_{ij} = |\widetilde{T}_{ij} - T_{ij}|. \tag{109}$$

For a predefined commit ratio p_{commit}, only positions satisfying

$$q_{ij} \geq Q_{1-p_{\mathrm{commit}}}(q) \tag{110}$$

are selected for discrete updates, where $Q_{1-p_{\mathrm{commit}}}(q)$ denotes the corresponding upper quantile computed over the currently active proxy entries.

The updated ternary matrix is obtained as

$$T_{ij}^{\mathrm{new}} = \begin{cases} T_{ij}^{\mathrm{cand}}, & q_{ij} \geq Q_{1-p_{\mathrm{commit}}}(q), \\ T_{ij}, & \text{otherwise.} \end{cases} \tag{111}$$

After each commit operation, the continuous proxy is reinitialized:

$$\widetilde{T} \leftarrow T^{\text{new}}, \tag{112}$$

which prevents the proxy from drifting away from the deployed discrete representation.

Memory-Efficient Proxy Activation. Maintaining continuous proxies for all ternary factors simultaneously introduces additional memory overhead. To address this issue, TATR adopts a stochastic module activation strategy. At each proxy-activation interval, a binary activation variable is sampled for each quantized linear matrix ℓ:

$$z_\ell \sim \text{Bernoulli}(p_{\text{act}}). \tag{113}$$

Only layers with $z_\ell = 1$ temporarily introduce ternary proxies:

$$\Theta^{(t)} = \Theta_{\text{scale}} \cup \{\widetilde{T}_{A,\ell}, \widetilde{T}_{B,\ell} | z_\ell = 1\}. \tag{114}$$

The remaining matrices continue scale-only calibration. Since proxy variables exist only during the corresponding optimization interval, TATR avoids maintaining additional full-model continuous copies.

Learning Rate Configuration. Because continuous scales and ternary proxies play different optimization roles, separate learning rates are adopted. The scale parameters use a conservative learning rate:

$$\eta_{\text{scale}}, \tag{115}$$

while ternary proxies use

$$\eta_{\text{ternary}} = \lambda_q \eta_{\text{scale}}, \qquad \lambda_q > 1. \tag{116}$$

The larger learning rate allows temporary proxies to accumulate sufficient evidence for discrete state changes within limited activation intervals.

Overall, TATR alternates between global scale calibration and sparse topology refinement. The final model discards all temporary proxies and only retains the hard ternary factors and continuous scales required for inference.

A.3 Effective Bit-Width of Quantized Linear Layers

Nominal quantization precision does not necessarily reflect the actual storage cost of a quantized model. This discrepancy becomes particularly significant for binary and other extremely low-bit PTQ methods, where auxiliary reconstruction parameters and metadata may constitute a non-negligible fraction of the overall model size. In addition to quantized weight codes, practical deployment may require residual binary components, BF16 scaling factors and means, group indicators, salient-weight masks, and structured sparsity indices. We therefore report the *effective bits per weight* (BPW), which explicitly accounts for all parameters required to reconstruct the quantized linear weights at inference time.

Throughout this appendix, BPW is computed only over the weight matrices of the quantized linear layers inside Transformer blocks. Full-precision embedding layers and LM heads are excluded from both the storage count and the denominator. Other non-weight components, including normalization parameters, biases, activations, and KV caches, are also excluded. Temporary quantities used only during calibration or optimization are not counted because they are discarded before deployment. Unless otherwise specified, all continuous reconstruction coefficients, such as scaling factors and means, are assumed to be stored in BF16, i.e., 16 bits per value.

For a linear weight matrix

$$W \in \mathbb{R}^{n \times m}, \tag{117}$$

where n and m denote the number of output and input channels, respectively, let M denote the total number of bits required by its deployable quantized representation. The effective BPW of this matrix is

$$\mathrm{BPW}(W) = \frac{M}{nm}. \tag{118}$$

For a model containing L quantized linear matrices, the model-level BPW is calculated by aggregating the actual storage over all quantized linear layers:

$$\mathrm{BPW}_{\mathrm{model}} = \frac{\sum_{\ell=1}^{L} M_\ell}{\sum_{\ell=1}^{L} n_\ell m_\ell}. \tag{119}$$

This parameter-weighted definition avoids bias toward small matrices and provides a consistent measure of the actual storage cost of the quantized Transformer linear layers.

GPTQ (Frantar et al., 2022). For GPTQ, we consider the 2-bit group-wise asymmetric quantization configuration used in our experiments. Each weight is represented by a 2-bit integer. Let g denote the group size and

$$q_g = \left\lceil \frac{m}{g} \right\rceil \tag{120}$$

the number of quantization groups along the input dimension for each output channel. Each group additionally stores one BF16 scaling factor and one 2-bit zero point. Therefore, the total storage of a GPTQ-quantized matrix is

$$M_{\mathrm{GPTQ}} = 2nm + (16+2)nq_g. \tag{121}$$

The corresponding effective BPW is

$$\mathrm{BPW}_{\mathrm{GPTQ}} = 2 + \frac{18q_g}{m}. \tag{122}$$

For the group size $g = 64$ used in our experiments, when m is divisible by 64,

$$\mathrm{BPW}_{\mathrm{GPTQ}} = 2 + \frac{18}{64} = 2.28125 \approx 2.28. \tag{123}$$

Hence, the effective storage cost is slightly higher than the nominal 2-bit representation because the group-wise scale and zero-point parameters must also be retained.

BiLLM (Huang et al., 2024). BiLLM separates each weight matrix into salient and non-salient columns and applies different binary representations to the two regions. Let c denote the number of salient columns, k the reconstruction block size, and

$$q = \left\lceil \frac{m}{k} \right\rceil \tag{124}$$

the number of column blocks per row.

For the salient columns, BiLLM adopts second-order residual binarization. Two binary tensors are stored for the original and residual components, requiring

$$2nc \tag{125}$$

bits. Each block further stores three BF16 row-wise reconstruction coefficients corresponding to the two scaling factors and the merged mean, which contributes

$$3 \times 16nq = 48nq \tag{126}$$

bits.

For the non-salient columns, first-order binarization stores one binary sign for each weight, resulting in

$$n(m-c) \tag{127}$$

bits. BiLLM further separates these weights into two distribution groups. Each group requires one BF16 scale and one BF16 mean, giving

$$2 \times 2 \times 16nq = 64nq \tag{128}$$

bits.

In addition, BiLLM requires a group bitmap over the full matrix and a salient-column bitmap. These respectively contribute nm and m bits. The complete storage is therefore

$$\begin{aligned} M_{\mathrm{BiLLM}} &= (2nc + 48nq) + \big[n(m-c) + 64nq\big] \\ &\quad + nm + m \\ &= 2nm + nc + m + 112nq. \end{aligned} \tag{129}$$

Dividing by the number of original weights gives

$$\mathrm{BPW}_{\mathrm{BiLLM}} = 2 + \frac{c}{m} + \frac{112q}{m} + \frac{1}{n}. \tag{130}$$

Therefore, although the primary weight representation is binary, the residual representation, reconstruction coefficients, and bitmap metadata increase the effective storage beyond the nominal 1-bit weight precision.

STBLLM$_{4:8}$ (Dong et al., 2024). STBLLM combines binary PTQ with structured $N:M$ sparsity. We consider the 4:8 configuration used in our experiments. As in BiLLM, the salient branch adopts second-order residual binarization and therefore requires

$$2nc + 48nq \tag{131}$$

bits.

For the non-salient branch, 4:8 structured sparsity retains four out of every eight weights. Consequently, only half of the corresponding binary signs need to be stored:

$$\frac{1}{2}n(m-c). \tag{132}$$

STBLLM further separates the non-salient weights into sparse, intermediate, and dense regions. The group assignment is represented using a 2-bit identifier for each retained entry, contributing

$$\frac{1}{2} \times 2nm = nm \tag{133}$$

bits.

Structured sparsity additionally requires the retained positions within each 4:8 block to be encoded. There are $\binom{8}{4} = 70$ possible patterns, and therefore

$$\left\lceil \log_2 \binom{8}{4} \right\rceil = 7 \tag{134}$$

bits are required for each eight-position block. The sparsity-index overhead is consequently

$$\frac{7}{8}n(m-c) \tag{135}$$

bits.

The three non-salient regions each require one BF16 scale and one BF16 mean, contributing

$$3 \times 2 \times 16nq = 96nq \tag{136}$$

bits. Finally, the salient-column bitmap requires an additional m bits. The total storage is

$$\begin{aligned} M_{\mathrm{STBLLM}_{4:8}} &= (2nc + 48nq) + \frac{1}{2}\big[n(m-c) + 2nm\big] \\ &\quad + \frac{7}{8}n(m-c) + 96nq + m \\ &= \frac{19}{8}nm + \frac{5}{8}nc + 144nq + m. \end{aligned} \tag{137}$$

Accordingly,

$$\mathrm{BPW}_{\mathrm{STBLLM}_{4:8}} = \frac{19}{8} + \frac{5c}{8m} + \frac{144q}{m} + \frac{1}{n}. \tag{138}$$

The effective BPW therefore reflects not only the retained binary weights, but also the structured sparsity indices, region assignments, reconstruction coefficients, and salient-weight information.

ARB-LLM$_{\mathrm{RC}}$ (Li et al., 2024). ARB-LLM$_{\mathrm{RC}}$ improves binary reconstruction by jointly introducing row-wise and column-wise continuous coefficients. The same salient-column notation c and block count q are used below.

For the salient columns, two binary residual components are stored, contributing

$$2nc \tag{139}$$

bits. Unlike purely row-wise reconstruction, ARB-LLM$_{\mathrm{RC}}$ additionally stores both row-wise and column-wise BF16 coefficients. The salient branch therefore requires

$$16(2nq + 2c) \tag{140}$$

bits for its continuous reconstruction parameters.

For the non-salient part, one binary sign is stored for each weight:

$$n(m - c). \tag{141}$$

Two groups of row-column reconstruction parameters are used, requiring

$$2 \times 16\big(nq + m - c\big) = 32\big(nq + m - c\big) \tag{142}$$

bits. ARB-LLM$_{\mathrm{RC}}$ additionally requires a group bitmap over the full matrix and a salient-column bitmap, contributing nm and m bits, respectively. Therefore,

$$\begin{aligned} M_{\mathrm{ARB_{RC}}} &= \big[2nc + 16(2nq + 2c)\big] \\ &\quad + \big[n(m - c) + 32(nq + m - c)\big] \\ &\quad + nm + m \\ &= 2nm + nc + 33m + 64nq. \end{aligned} \tag{143}$$

The resulting effective BPW is

$$\mathrm{BPW}_{\mathrm{ARB_{RC}}} = 2 + \frac{c}{m} + \frac{64q}{m} + \frac{33}{n}. \tag{144}$$

The additional row-column reconstruction coefficients improve the flexibility of the binary approximation but also contribute to the effective storage cost.

HBLLM$_{\mathrm{row}}$ (Chen et al., 2025). HBLLM$_{\mathrm{row}}$ extends binary PTQ with Haar-domain row-wise quantization and frequency-aware grouping. Its practical storage differs from a standard salient/non-salient binary representation because the non-salient Haar branch uses FillAvg to fill the salient positions before row-wise Haar-domain quantization.

As a result, the non-salient Haar branch operates on and stores a dense binary representation over all nm positions rather than only the original $n(m - c)$ non-salient entries. This binary representation therefore requires

$$nm \tag{145}$$

bits. The row-wise Haar branch uses two frequency-aware groups, each associated with three BF16 reconstruction coefficients, resulting in

$$3 \times 16nq \times 2 = 96nq \tag{146}$$

bits.

For the salient branch, nc binary signs are stored. Two groups of BF16 scale/mean parameters further require

$$2 \times 2 \times 16nq = 64nq \tag{147}$$

bits.

In addition, the FillAvg-based row branch requires a group bitmap covering both the original matrix and the filled salient positions, resulting in

$$n(m + c) \tag{148}$$

bits. The salient-column bitmap contributes another m bits. Hence,

$$\begin{aligned} M_{\mathrm{HBLLM_{row}}} &= (nm + 96nq) + (nc + 64nq) \\ &\quad + n(m + c) + m \\ &= 2nm + 2nc + m + 160nq. \end{aligned} \tag{149}$$

Therefore,

$$\mathrm{BPW}_{\mathrm{HBLLM_{row}}} = 2 + \frac{2c}{m} + \frac{160q}{m} + \frac{1}{n}. \tag{150}$$

This formulation accounts for the dense binary Haar representation, frequency-aware reconstruction coefficients, salient branch, and the corresponding bitmap metadata.

TWLA (Zhao et al., 2026). TWLA performs post-training ternarization using the asymmetric E2M-ATQ representation

$$\bar{W} = \boldsymbol{\mu}\mathbf{1}^\top + \mathrm{Diag}(\boldsymbol{\alpha})T, \qquad T \in \{-1, 0, +1\}^{n\times m}, \tag{151}$$

where $\boldsymbol{\mu}, \boldsymbol{\alpha} \in \mathbb{R}^n$ denote the row-wise shift and scale, respectively. Although the ternary codebook corresponds to a nominal precision of

$$b_{\mathrm{ter}} = \log_2 3 \approx 1.585 \tag{152}$$

bits per weight, the deployable representation requires additional metadata and reconstruction parameters.

Following the implementation inherited from ARB-LLM, the quantized weights are further associated with importance-aware partition information. Let s denote the number of weights assigned to the important region. A group bitmap over the full matrix requires nm bits, while the salient-column indicator requires m bits. In addition, important weights retain one extra bit for the higher-fidelity representation, contributing s bits. The two row-wise BF16 reconstruction vectors $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$ require another

$$2 \times 16n = 32n \tag{153}$$

bits.

Therefore, excluding activation quantization and other activation-side information, the effective weight storage of a TWLA-quantized linear matrix is

$$M_{\mathrm{TWLA}} = b_{\mathrm{ter}}nm + nm + s + m + 32n. \tag{154}$$

The corresponding effective BPW is

$$\mathrm{BPW}_{\mathrm{TWLA}} = b_{\mathrm{ter}} + 1 + \frac{s}{nm} + \frac{1}{n} + \frac{32}{m}. \tag{155}$$

When importance is assigned column-wise, let c denote the number of important columns. Since $s = nc$, Eq. 155 reduces to

$$\mathrm{BPW}_{\mathrm{TWLA}} = \log_2 3 + 1 + \frac{c}{m} + \frac{1}{n} + \frac{32}{m}. \tag{156}$$

For the complete model, the effective BPW is obtained using the same parameter-weighted aggregation as Eq. 119:

$$\mathrm{BPW}^{\mathrm{model}}_{\mathrm{TWLA}} = \frac{\sum_{\ell=1}^{L} [(\log_2 3)n_\ell m_\ell + n_\ell m_\ell + s_\ell + m_\ell + 32n_\ell]}{\sum_{\ell=1}^{L} n_\ell m_\ell}. \tag{157}$$

Using the importance masks and layer configurations adopted in our comparison gives an effective weight cost of approximately 2.92 BPW. This value is higher than the nominal W1.58 designation because the latter describes only the ternary codebook, whereas our effective BPW additionally accounts for importance metadata, the extra representation bit for important weights, and the continuous reconstruction parameters.

Ternova (ours). Ternova represents each dense linear weight matrix using two low-rank ternary factors rather than a dense ternary matrix. For

$$W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}, \tag{158}$$

the final deployable representation is

$$\widehat{W} = \operatorname{Diag}(a) T_A \operatorname{Diag}(m) T_B \operatorname{Diag}(b), \tag{159}$$

where

$$T_A \in \{-1, 0, +1\}^{d_{\text{out}} \times r}, \qquad T_B \in \{-1, 0, +1\}^{r \times d_{\text{in}}}, \tag{160}$$

and r denotes the rank assigned by SRMA. The three continuous scaling vectors are

$$a \in \mathbb{R}^{d_{\text{out}}}, \qquad m \in \mathbb{R}^{r}, \qquad b \in \mathbb{R}^{d_{\text{in}}}. \tag{161}$$

For practical storage, five ternary values can be encoded in one 8-bit integer because

$$3^5 = 243 < 256. \tag{162}$$

Thus, the two ternary factors require

$$M_{\text{ter}} = 8 \left\lceil \frac{d_{\text{out}} r}{5} \right\rceil + 8 \left\lceil \frac{r d_{\text{in}}}{5} \right\rceil \tag{163}$$

bits.

In our implementation, SRMA controls the mixed ranks such that the parameter-weighted storage of the two packed ternary factors is approximately 1.58 BPW:

$$\frac{\displaystyle\sum_{\ell=1}^{L} \left[8 \left\lceil \frac{d_{\text{out},\ell} r_\ell}{5} \right\rceil + 8 \left\lceil \frac{r_\ell d_{\text{in},\ell}}{5} \right\rceil \right]}{\displaystyle\sum_{\ell=1}^{L} d_{\text{out},\ell} d_{\text{in},\ell}} \approx 1.58. \tag{164}$$

The above 1.58 BPW accounts only for the packed ternary factors. The three continuous scaling vectors are additionally stored in BF16 and therefore require

$$M_{\text{scale}} = 16 \left(d_{\text{out}} + r + d_{\text{in}} \right) \tag{165}$$

bits for each linear layer.

Accordingly, the complete storage of a Ternova-compressed linear layer is

$$\begin{aligned} M_{\text{Ternova}} = {} & 8 \left\lceil \frac{d_{\text{out}} r}{5} \right\rceil + 8 \left\lceil \frac{r d_{\text{in}}}{5} \right\rceil \\ & + 16 \left(d_{\text{out}} + r + d_{\text{in}} \right). \end{aligned} \tag{166}$$

For the complete model, the additional BPW introduced by the three BF16 scaling vectors is

$$\text{BPW}_{\text{scale}} = \frac{16 \displaystyle\sum_{\ell=1}^{L} \left(d_{\text{out},\ell} + r_\ell + d_{\text{in},\ell} \right)}{\displaystyle\sum_{\ell=1}^{L} d_{\text{out},\ell} d_{\text{in},\ell}}. \tag{167}$$

Therefore, the effective model-level BPW of Ternova is

$$\boxed{\text{BPW}_{\text{Ternova}} = 1.58 + \text{BPW}_{\text{scale}}} \tag{168}$$

where the first term corresponds to the two packed ternary factors and the second term accounts for all three BF16 scaling vectors.

For intuition, consider a 4096×4096 linear matrix. A rank of approximately $r = 2022$ results in a packed ternary-factor cost of approximately 1.58 BPW. The three BF16 scaling vectors introduce

$$\frac{16(4096 + 2022 + 4096)}{4096^2} \approx 0.0097 \tag{169}$$

additional BPW, resulting in a total effective storage of approximately 1.59 BPW.

The Hessian-aware statistics used by HATF, the sensitivity information used by SRMA, and the topology proxies introduced during TATR are required only during calibration and are discarded before deployment. They therefore introduce no additional deployment storage.

A.4 Additional Experimental Details

A.4.1 Experimental Hyperparameter Settings

HATF. For HATF, we use a unified set of hyperparameters across all model families unless otherwise specified. In Stage II, the TSVID-guided ADMM factorization follows the alternating-optimization framework of DBF (Boža & Macko, 2025). We perform 260 outer alternating iterations, with 3 inner ADMM updates for each factor update, and set the ADMM regularization coefficient to 3×10^{-2}. For each TSVID projection, the leading magnitude component is initialized using 5 power iterations. The adaptive ternary threshold is fixed to $\theta = 0.5$, corresponding to the nearest-code decision boundary. We then perform 3 TSVID refinement iterations, with 3 alternating least-squares updates of the row- and column-wise scaling vectors in each refinement iteration.

In Stage III, we follow the annealed discrete optimization setting of GSQ (Dadgarnia et al., 2026) and adapt it to the mask–sign parameterization of our ternary factors. The mask and sign logits are optimized using Lion with a learning rate of 1×10^{-4}, weight decay of 1.0, and momentum coefficients $(0.9, 0.95)$. The temperature is linearly annealed from $\tau_{\max} = 2.0$ to $\tau_{\min} = 0.05$, while the logit amplification coefficient is linearly increased from $\kappa_{\min} = 100$ to $\kappa_{\max} = 500$. The initialization noise standard deviation is set to $\sigma_{\text{init}} = 0.01$. We set the base mask- and sign-regularization strengths to 0.03; these base strengths are modulated by the time-dependent hardening schedules described in Appendix A.2.4. Unlike GSQ, the continuous scaling vectors $\{a, m, b\}$ are not optimized by an additional gradient-based optimizer; instead, they are periodically refitted using the closed-form least-squares updates described in Appendix A.2.5. All annealing schedules are applied linearly over the Stage-III refinement process.

SRMA. For SRMA, we first construct a higher-capacity uniform-rank HATF probe with an internal ternary-factor BPW budget of 2.0, which provides a sufficiently rich set of rank components for subsequent importance estimation and mixed-rank allocation. We set the sensitivity exponent to $\gamma = 0.5$. Rank allocation is performed in blocks of $q = 16$, with a minimum rank $r_{\min} = 256$ and a relative rank-floor coefficient of $\eta = 0.8$. Matrix-level sensitivity is estimated using 256 calibration sequences. Starting from the resulting rank floors, rank blocks are greedily allocated according to their storage-normalized allocation density until the target BPW budget is reached.

TATR. For TATR, we perform token-level teacher–student distillation while keeping the mixed-rank configuration produced by SRMA fixed. The continuous scale vectors are optimized using BF16FusedAdamW with a base learning rate of 1×10^{-5}, momentum coefficients $(0.9, 0.95)$, and weight decay of 1×10^{-4}. Following the implementation in our scale calibration, the learning rate of each scale vector is further proportional to its ℓ_2 norm.

For sparse ternary topology refinement, temporary STE proxies are optimized with a learning-rate multiplier of $\lambda_q = 300$, corresponding to a proxy learning rate of 3×10^{-3}, with momentum coefficients $(0, 0.95)$. The refinement is performed for 256 optimization steps with a batch size of 16. Every 50 steps, each quantized linear module is independently sampled for the next topology-refinement interval with probability $p_{\text{act}} = 0.1$. Newly activated proxies are warmed up for 5 optimization steps before discrete updates are committed. At each commit, only approximately the top 0.001% highest-confidence ternary transitions are committed, corresponding to $p_{\text{commit}} = 10^{-5}$. All discrete updates are restricted to adjacent ternary transitions $-1 \leftrightarrow 0 \leftrightarrow +1$.

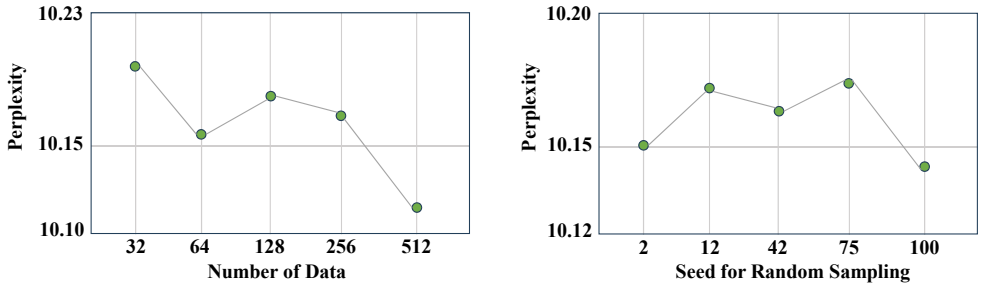
A.4.2 Detailed Main Results

Table 5 expands the averaged zero-shot results reported in Table 1 into task-level accuracy on ARC-Easy, ARC-Challenge, HellaSwag, LAMBADA-openai, LAMBADA-standard, PIQA, and Wino-Grande. Despite using only 1.59 BPW, Ternova achieves the highest average zero-shot accuracy on seven of the eight evaluated models, including both LLaMA-3 models, both Gemma-3 models, Qwen3-14B, and both Qwen3.5 models. On Qwen3-8B, Ternova remains highly competitive, achieving 61.41 average accuracy compared with 61.78 for TWLA, while operating at a substantially lower effective bit-width. The task-level breakdown further shows that the improvement is not driven by an isolated benchmark: Ternova consistently attains strong or leading performance across commonsense reasoning, language modeling, and cloze-style completion tasks. In particular, it achieves the best or second-best quantized result on the majority of individual tasks across

the evaluated model families, indicating that the proposed HATF, SRMA, and TATR components preserve broad task-level capability rather than overfitting to a specific benchmark. These detailed results are consistent with the averaged accuracy and WikiText2 perplexity reported in Table 1, further demonstrating the effectiveness of Ternova under an extremely constrained weight budget.

Table 5: Detailed zero-shot accuracy on ARC-Easy (AE), ARC-Challenge (AC), HellaSwag (HS), LAMBADA-openai (LO), LAMBADA-standard (LS), PIQA (PQ), and WinoGrande (WG). The average score is computed over the seven zero-shot tasks. Bold and underline denote the best and second-best quantized results within each model.

Model	Method	BPW	AE↑	AC↑	HS↑	LO↑	LS↑	PQ↑	WG↑	Avg.[7] ↑
LLaMA-3-8B	BF16	16	77.90	52.82	79.07	75.63	68.58	80.63	72.93	72.51
	BiLLM	2.88	36.78	22.18	34.81	28.76	13.00	58.49	53.67	35.38
	STBLLM$_{(4:8)}$	3.50	31.10	21.76	31.29	11.84	5.88	54.52	49.64	29.43
	ARB-LLM$_{RC}$	2.51	48.78	28.33	43.55	50.65	29.83	65.61	56.75	46.21
	HBLLM$_{row}$	3.25	54.12	32.68	60.22	58.51	46.40	70.35	**65.19**	55.35
	TWLA	2.92	61.20	35.80	63.70	61.20	51.00	72.20	62.65	58.25
	Ternova	**1.59**	**63.97**	**38.65**	**66.28**	**64.45**	**54.08**	**74.97**	64.40	**60.97**
LLaMA-3-70B	BF16	16	86.07	64.33	84.92	79.31	73.96	84.60	80.35	79.08
	BiLLM	2.88	25.34	20.73	33.18	9.41	5.44	52.39	51.78	28.32
	STBLLM$_{(4:8)}$	3.50	25.67	25.94	27.23	1.84	4.29	51.03	49.88	26.55
	ARB-LLM$_{RC}$	2.51	61.71	42.48	67.44	65.68	53.54	69.97	64.53	60.76
	HBLLM$_{row}$	3.25	24.83	27.05	**74.70**	70.66	50.96	50.60	**73.56**	53.19
	TWLA	2.92	71.10	50.85	74.20	70.85	61.45	75.60	71.60	67.95
	Ternova	**1.59**	**71.64**	**51.39**	74.57	**71.22**	**61.87**	**75.94**	70.98	**68.23**
Gemma-3-4B-it	BF16	16	78.20	56.74	74.09	59.17	54.49	77.15	70.01	67.12
	BiLLM	2.88	36.57	24.91	35.29	9.33	6.64	60.01	53.59	32.33
	STBLLM$_{(4:8)}$	3.50	33.88	22.18	31.34	5.67	4.68	55.28	52.17	29.31
	ARB-LLM$_{RC}$	2.51	49.20	29.10	40.62	30.35	23.33	65.61	54.62	41.83
	HBLLM$_{row}$	3.25	**60.40**	**36.52**	**54.13**	41.68	33.01	**69.42**	57.30	50.35
	TWLA	2.92	58.40	31.60	51.20	45.80	39.70	68.00	**60.83**	50.79
	Ternova	**1.59**	59.85	33.11	52.18	**47.06**	**41.80**	69.26	58.01	**51.61**
Gemma-3-12B-it	BF16	16	76.73	60.32	81.89	67.71	65.61	78.51	74.66	72.20
	BiLLM	2.88	43.60	28.67	41.82	22.06	13.97	62.79	55.96	38.41
	STBLLM$_{(4:8)}$	3.50	34.01	24.06	33.57	13.45	8.52	54.68	51.93	31.46
	ARB-LLM$_{RC}$	2.51	46.93	29.27	45.98	50.73	34.60	64.64	57.93	47.15
	HBLLM$_{row}$	3.25	67.80	**43.69**	**67.00**	43.92	37.61	**74.70**	**67.09**	57.40
	TWLA†	2.92	66.80	41.90	62.70	55.50	52.90	72.60	62.91	59.33
	Ternova	**1.59**	**68.35**	43.34	64.18	**57.00**	**54.65**	73.78	65.67	**60.99**
Qwen3-8B	BF16	16	80.93	56.74	74.98	64.18	61.11	77.37	68.35	69.09
	BiLLM	2.88	34.97	23.81	37.69	18.75	12.71	56.80	51.14	33.70
	STBLLM$_{(4:8)}$	3.50	31.78	22.61	29.89	5.72	2.99	53.81	49.33	28.02
	ARB-LLM$_{RC}$	2.51	59.09	36.35	49.84	41.86	34.29	68.72	58.80	49.85
	HBLLM$_{row}$	3.25	58.38	36.18	60.51	50.32	44.91	71.82	61.17	54.76
	TWLA	2.92	**72.80**	**45.90**	**65.20**	**57.10**	**52.70**	**73.00**	**65.76**	**61.78**
	Ternova	**1.59**	72.22	45.56	64.91	56.78	52.53	72.25	65.59	61.41
Qwen3-14B	BF16	16	83.08	60.49	78.82	67.84	64.47	79.76	72.85	72.47
	BiLLM	2.88	56.57	34.56	55.70	48.67	43.90	70.62	64.72	53.53
	STBLLM$_{(4:8)}$	3.50	47.52	27.47	41.33	24.72	22.86	63.06	56.20	40.88
	ARB-LLM$_{RC}$	2.51	72.43	44.28	60.55	61.50	51.85	73.88	66.69	61.60
	HBLLM$_{row}$	3.25	69.19	46.93	69.60	**63.61**	56.53	74.92	68.03	64.12
	TWLA	2.92	76.60	53.70	72.40	62.30	57.60	77.50	**71.35**	67.35
	Ternova	**1.59**	**77.31**	**54.44**	**73.24**	63.28	**58.49**	**78.24**	69.61	**67.80**
Qwen3.5-9B	BF16	16	74.07	55.63	78.09	70.21	63.40	80.36	72.93	70.67
	BiLLM	2.88	56.06	30.38	48.87	47.89	41.76	66.43	58.56	49.99
	STBLLM$_{(4:8)}$	3.50	47.05	26.96	41.86	35.44	28.37	63.38	54.22	42.47
	ARB-LLM$_{RC}$	2.51	64.10	36.01	53.34	53.31	46.67	70.40	61.33	55.02
	HBLLM$_{row}$	3.25	72.26	46.42	67.84	**65.11**	**57.46**	74.37	**68.75**	64.60
	TWLA	2.92	75.50	49.80	67.20	58.50	53.40	74.20	64.29	63.27
	Ternova	**1.59**	**77.95**	**52.47**	**69.52**	61.77	56.70	**76.61**	68.03	**66.15**
Qwen3.5-27B	BF16	16	79.84	61.09	83.40	75.16	71.26	82.26	78.77	75.97
	BiLLM	2.88	72.01	48.21	68.17	73.32	66.25	75.35	72.45	67.96
	STBLLM$_{(4:8)}$	3.50	69.87	43.86	63.77	69.75	62.08	73.34	70.56	64.74
	ARB-LLM$_{RC}$	2.51	77.95	52.39	70.12	**75.24**	**70.52**	77.91	72.85	70.99
	HBLLM$_{row}$	3.25	78.45	55.63	78.20	75.00	68.66	80.41	**75.69**	73.14
	TWLA	2.92	83.90	61.90	77.10	69.10	64.50	80.90	73.18	72.94
	Ternova	**1.59**	**85.65**	**64.85**	**78.66**	71.08	66.58	**82.10**	73.40	**74.62**



(a) Change of Number **(b) Change of Seed**

Figure 4: Ablation on calibration data for TERNOVA on Qwen3.5-9B. We use RedPajama as the calibration corpus and report WikiText2 perplexity after ternarization. (a) Varying the number of calibration samples. (b) Varying the random seed for calibration-data sampling at a fixed calibration-set size. Lower perplexity is better.

A.4.3 ABLATION ON CALIBRATION DATA

We investigate the sensitivity of TERNOVA to calibration-data configurations on Qwen3.5-9B. Following our experimental setup, we sample calibration sequences from RedPajama, each containing 2,048 tokens, and evaluate the ternarized model using WikiText2 perplexity. We examine two factors: calibration-set size and calibration-sampling randomness, while keeping the remaining quantization hyperparameters unchanged. Specifically, we vary the number of calibration samples over $\{32, 64, 128, 256, 512\}$. For the seed-sensitivity study, we fix the calibration-set size and use sampling seeds from $\{2, 12, 42, 75, 100\}$.

As shown in Fig. 4(a), TERNOVA maintains stable performance across the tested calibration sizes, with a perplexity spread of less than 0.1. The relationship between calibration size and perplexity is not strictly monotonic. Although 512 samples yield the lowest perplexity, the improvement over 256 samples is only approximately 0.05. These results suggest that increasing the calibration-set size provides limited gains in this setting, while smaller calibration sets already yield comparable performance. Similarly, Fig. 4(b) shows only minor fluctuations across sampling seeds, with a maximum perplexity difference below 0.04. This suggests that the observed performance is not strongly dependent on a particular sampled subset. Taken together, these results demonstrate robustness to the tested calibration-set sizes and sampling seeds on Qwen3.5-9B, supporting the use of a modest calibration set for post-training ternarization.

A.4.4 COMPARISON WITH DBF

Methodological differences. DBF (Boža & Macko, 2025) approximates each weight matrix using two binary factors and continuous scaling vectors. It incorporates input/output importance scaling into alternating ADMM optimization with SVID projections, and supports nonuniform intermediate dimensions through gradient-based channel-importance scoring. Its PV-tuned variant, denoted DBF+PV, further refines binary signs alongside continuous parameters.

TERNOVA builds on this scaled-factorization formulation but differs in three aspects. First, HATF introduces TSVID-guided ternary factorization and annealed joint refinement, explicitly optimizing both zero/nonzero assignments and signs while fitting continuous scales separately. Second, SRMA evaluates rank-component importance through refit-aware deletion loss, which measures the reconstruction damage after removing a component and refitting the remaining coefficients. By combining this criterion with matrix-level task sensitivity and storage cost, SRMA accounts for both compensation among components and heterogeneous sensitivity across matrices. Third, TATR combines global scale calibration with sparse, high-confidence ternary updates restricted to adjacent state transitions ($-1 \leftrightarrow 0 \leftrightarrow +1$), while keeping the allocated ranks fixed. These differences extend beyond the discrete value set to the optimization of factor representations, the allocation of rank capacity, and the mechanism of task-level refinement.

Experimental comparison. To directly assess the two frameworks under the same storage constraint, we compare DBF+PV and TERNOVA on LLaMA-3-8B and Qwen3.5-9B at 1.59 effective BPW. For both methods, this budget includes the packed discrete weight factors and all associated scaling vectors, normalized by the number of original weights in the quantized linear layers. Full-precision embeddings and the LM head are excluded from this accounting. For TERNOVA, we retain the default higher-capacity HATF probe and control the final mixed ranks through SRMA to match the target budget. We report WikiText2 perplexity and average accuracy over the seven zero-shot tasks used in the main experiments.

Table 6: Comparison with DBF+PV under a matched storage budget. Both compressed methods use 1.59 effective BPW, including the discrete weight factors and scaling vectors of the quantized linear layers. PPL is measured on WikiText2, and Avg. denotes average accuracy (%) over seven zero-shot tasks. BF16 results are included for reference. Bold indicates the best quantized result.

Method	BPW	LLaMA-3-8B		Qwen3.5-9B	
		PPL↓	Avg.↑	PPL↓	Avg.↑
BF16	16	6.14	72.51	8.65	70.67
DBF+PV	1.59	12.48	57.83	11.24	64.05
TERNOVA	1.59	**10.94**	**60.97**	**10.17**	**66.15**

As shown in Table 6, TERNOVA outperforms DBF+PV on both models under the matched storage budget. On LLaMA-3-8B, it reduces perplexity from 12.48 to 10.94, a relative reduction of 12.3%, while improving average zero-shot accuracy from 57.83% to 60.97%, a gain of 3.14 percentage points. On Qwen3.5-9B, perplexity decreases from 11.24 to 10.17, corresponding to a 9.5% relative reduction, and average zero-shot accuracy increases from 64.05% to 66.15%, a gain of 2.10 percentage points. These results demonstrate that TERNOVA achieves better language-modeling and zero-shot performance than PV-refined binary factorization without increasing the effective storage budget of the compressed linear weights.

A.4.5 DETAILED ABLATION OF TASK-ALIGNED TERNARY REFINEMENT

We further investigate the individual contributions of global scale calibration and sparse topology refinement within TATR on Qwen3.5-9B. All variants retain HATF and SRMA and operate at 1.59 effective BPW, including the ternary weight factors and their scaling vectors. The mixed-rank configuration remains fixed during refinement. We compare four configurations: *without TATR*, which directly evaluates the model after HATF and SRMA; *scale-only*, which optimizes the continuous scaling vectors while freezing all ternary factors; *topology-only*, which performs sparse ternary-state updates while keeping the existing scaling vectors fixed; and *full TATR*, which enables both mechanisms. We report WikiText2 perplexity and average accuracy over the seven zero-shot tasks used in the main experiments.

Table 7: Detailed ablation of TATR on Qwen3.5-9B at 1.59 effective BPW. All variants retain HATF and SRMA. The scale and topology columns indicate whether the corresponding updates are enabled during TATR; disabling scale calibration freezes rather than removes the existing scales. PPL is measured on WikiText2, and Avg. denotes average accuracy (%) over seven zero-shot tasks. Bold indicates the best results.

Variant	Scale calibration	Topology refinement	PPL↓	Avg.↑
Without TATR	×	×	12.75	60.79
Scale-only	✓	×	11.27	64.21
Topology-only	×	✓	11.83	63.09
Full TATR	✓	✓	**10.17**	**66.15**

As shown in Table 7, both mechanisms improve performance when applied individually. Scale-only calibration reduces perplexity from 12.75 to 11.27 and increases average zero-shot accuracy from 60.79% to 64.21%, indicating that model-level magnitude adjustment can improve the existing ternary representation without modifying its discrete states. Topology-only refinement also improves perplexity to 11.83 and average accuracy to 63.09%, showing that task-guided discrete-state updates provide benefits even when the continuous scales are frozen. Although scale-only achieves better results than topology-only in this setting, neither variant matches the complete refinement procedure.

Combining both mechanisms yields the best perplexity of 10.17 and average accuracy of 66.15%. Compared with scale-only calibration, full TATR further reduces perplexity by 1.10 and improves average accuracy by 1.94 percentage points, demonstrating that its gains are not attributable to scale recalibration alone. Overall, full TATR lowers perplexity by 2.58 and increases average accuracy by 5.36 percentage points relative to the model without TATR. These results support the complementary roles of continuous scale calibration and discrete topology refinement in improving task-level performance while preserving the deployment storage budget.

A.4.6 Analysis of Mixed-Rank Allocation

To examine how SRMA distributes representational capacity, we analyze its rank allocation on Qwen3.5-27B. Starting from a higher-capacity HATF probe, SRMA evaluates the refit-aware importance of candidate rank components and combines it with matrix-level task sensitivity. It then directly determines the mixed-rank configuration through cost-aware block allocation under a target factor budget of 1.58 BPW. Specifically, structurally related matrices are organized into candidate groups, and feasible rank blocks are selected according to their storage-normalized utility within each group. We use a sensitivity exponent of $\gamma = 0.5$, a rank-block size of $q = 16$, a minimum rank of $r_{\min} = 256$, and a relative rank-floor coefficient of $\eta = 0.8$.

We summarize the resulting configuration by four matrix groups: `linear_attn`, `kv`, `qo`, and `mlp`. For each group, Table 8 reports the number of quantized matrices, the minimum, mean, and maximum assigned ranks, and the group's contribution to the total ternary-factor entry count. These statistics characterize both the matrix-level latent dimensions and the distribution of factor entries across groups. For a matrix u, the two ternary factors contain $r_u(d_{\mathrm{out},u} + d_{\mathrm{in},u})$ entries. Consequently, a group's factor-entry share depends on its matrix count, matrix dimensions, and assigned ranks. The rank distribution should therefore be interpreted together with matrix geometry, rather than treating larger raw ranks or entry shares as direct evidence of higher task sensitivity.

The performance improvement of SRMA over individual allocation criteria comes from the joint consideration of complementary signals. Task sensitivity identifies matrices whose perturbations have larger effects on model-level behavior, while the refit-aware deletion loss evaluates the marginal replaceability of individual rank components after compensating the remaining factors. Furthermore, the storage-normalized allocation density prevents the rank budget from being dominated by large matrices and favors components with higher utility per additional storage cost. Therefore, SRMA performs budget-constrained capacity allocation rather than simply assigning ranks according to reconstruction error or sensitivity alone.

This structural analysis complements the performance comparison of rank-allocation strategies in the main text.

Table 8: Group-level statistics of the provided mixed-rank configuration for Qwen3.5-27B. Count denotes the number of matrix-level rank assignments, rather than Transformer blocks. Min., Mean, and Max. summarize the assigned ranks within each group. Entries denotes the number of elements in both ternary factors, and Share denotes each group's proportion of all factor entries.

Group	Count	Assigned rank			Entries (10^9)	Share (%)
		Min.	Mean	Max.		
`linear_attn`	240	256	1,901.6	6,160	5.6534	23.09
`kv`	32	704	863.0	1,504	0.1697	0.69
`qo`	32	2,240	3,405.5	5,040	1.5138	6.18
`mlp`	192	3,184	3,963.5	7,152	17.1436	70.03
All groups	496	256	2,729.8	7,152	24.4805	100.00

A.5 Qualitative Generation Examples

To illustrate the qualitative impact of ultra-low-bit compression, Table 9 compares selected generations from Qwen3.5-27B using the BF16 baseline, BiLLM, STBLLM, ARB-LLM, HBLLM, and our proposed TERNOVA. Rather than open-ended conversational prompts, we consider four short questions with objective answers, covering sequential percentage calculations, constrained task ordering, average-speed computation, and deductive reasoning. These examples allow us to examine both the preservation of correct answer content and the coherence of the generated explanations.

The assessment column distinguishes answer correctness from observable generation failures, including empty outputs, prompt copying, repetitive continuations, and token degeneration. In particular, a response containing the correct answer is not treated as fully satisfactory when it repeatedly reproduces the same statement. Dark green highlights correct key information, dark red marks incorrect answers or generation failures, and dark orange identifies correct but unstable outputs. Repeated continuations are abbreviated for readability. These annotations provide qualitative context for the displayed responses rather than replacing standard benchmark evaluation.

As shown in Table 9, the baseline methods exhibit distinct failure modes on the selected prompts. BiLLM produces no response to the discount-and-tax question (Q1), repeats the correct ordering in Q2, and degenerates into repeated token strings in the average-speed question (Q3). STBLLM yields empty outputs for Q1 and Q3, copies the prompt without answering Q2, and produces an incorrect, repetitive response to the deductive-reasoning question (Q4). ARB-LLM and HBLLM correctly answer Q1, Q2, and Q4, but both fail on Q3 in different ways: ARB-LLM incorrectly sums the two travel distances as 150 km and consequently reports 50 km/h, whereas HBLLM produces an empty output. These cases illustrate that preserving correct answers on some prompts does not necessarily prevent arithmetic errors or generation failures on others.

In contrast, TERNOVA provides the correct key answer for all four selected prompts, accompanied by concise and coherent explanations. It correctly applies the discount before taxation to obtain \$194.40, satisfies the ordering constraints with $A \to D \to C \to B$, computes the average speed as total distance divided by total travel time to obtain 70 km/h, and derives the correct negative conclusion in Q4. Across these examples, its answers agree with the BF16 baseline, without the repetition or token degeneration observed in several baseline outputs. These observations illustrate the preservation of basic arithmetic and logical reasoning under ultra-low-bit ternarization and complement the quantitative results in the main text. Since the table contains only a small set of selected examples, it should not be interpreted as a comprehensive assessment of reasoning ability or an estimate of generation failure rates.

Table 9: Qualitative comparison of selected Qwen3.8-27B generations under BF16 and ultra-low-bit PTQ methods. The four prompts have objective answers and cover arithmetic, constrained ordering, and deductive reasoning. Dark green highlights correct key information, dark red marks incorrect answers or generation failures, and dark orange denotes correct but unstable outputs. Repeated continuations are abbreviated for readability. These examples complement, rather than replace, quantitative benchmark evaluation.

Prompt	Method	Response	Assessment
Q1: A jacket originally costs \$240. It is discounted by 25%, and then an 8% sales tax is applied to the discounted price. What is the final price? **Answer:** \$194.40	BF16	The discounted price is \$180. Applying an 8% sales tax gives \$180 + \$14.40 = \$194.40.	Correct
	BiLLM	No response is generated.	Empty output
	STBLLM	No response is generated.	Empty output
	ARB-LLM	The discounted price is \$180. The sales tax is \$14.40, giving a final price of \$194.40.	Correct
	HBLLM	\$240 × 0.25 = \$60, so the discounted price is \$180. The final price is \$180 + \$14.40 = \$194.40.	Correct
	Ternova	The discounted price is \$240 × 0.75 = \$180. Applying the 8% tax gives \$194.40.	Correct
Q2: Four tasks A, B, C, and D must be completed in some order. A must be before C, D must be immediately after A, and B must be last. What is the only possible order? **Answer:** A, D, C, B	BF16	Since B must be last and D must immediately follow A, the only valid arrangement is A, D, C, B.	Correct
	BiLLM	The order must be A, D, C, B. "The order must be A, D, C, B" ··· repeated.	Correct but repetitive
	STBLLM	Four tasks A, B, C, and D must be completed in some order. A must be before C, D must be immediately after A, and B must be last.	Prompt copying
	ARB-LLM	D must immediately follow A, and B must be last. Therefore, the only possible order is A → D → C → B.	Correct
	HBLLM	A and D form a fixed pair, B is last, and C occupies the remaining position. Thus, the order is A, D, C, B.	Correct
	Ternova	A must precede C, D must immediately follow A, and B must be last. Hence, the only possible sequence is A, D, C, B.	Correct
Q3: A car travels 90 km at 60 km/h and then 120 km at 80 km/h. What is its average speed over the entire trip? **Answer:** 70 km/h	BF16	The two segments each take 1.5 hours. Thus, the total distance is 210 km and the total time is 3 hours, giving 70 km/h.	Correct
	BiLLM	</think> ··· </think>	Degenerated
	STBLLM	No response is generated.	Empty output
	ARB-LLM	Total distance = 90 + 120 = 150 km. The total time is 3 hours, so the average speed is 50 km/h.	Incorrect
	HBLLM	No response is generated.	Empty output
	Ternova	The two segments each take 1.5 hours. The total distance is 210 km and the total time is 3 hours, yielding 70 km/h.	Correct
Q4: All Zorps are Mips. No Mips are Lats. Can any Zorp be a Lat? Answer yes or no and briefly explain. **Answer:** No	BF16	Since all Zorps are Mips and no Mips are Lats, no Zorp can be a Lat.	Correct
	BiLLM	No. If all Zorps are Mips and no Mips are Lats, then no Zorp can be a Lat.	Correct
	STBLLM	Yes, a Zorp can be a Lat. "Zorp → Mip → Lat" ··· repeated.	Wrong + repetitive
	ARB-LLM	No. Since all Zorps are Mips and no Mips are Lats, Zorps cannot be Lats.	Correct
	HBLLM	No. Since all Zorps are Mips, and no Mips are Lats, it is impossible for any Zorp to be a Lat.	Correct
	Ternova	No. Every Zorp is a Mip, while no Mip can be a Lat. Therefore, no Zorp can be a Lat.	Correct