Sherry: Hardware-Efficient 1.25-Bit Ternary Quantization via Fine-grained Sparsification

Hong Huang[1,2] **Decheng Wu**[2] **Qiangqiang Hu**[2] **Guanghua Yu**[2] **Jinhai Yang**[1]
Jianchen Zhu[2] **Xue Liu**[3] **Dapeng Wu**[1]
[1]City University of Hong Kong [2]Tencent [3]McGill University

Abstract

The deployment of Large Language Models (LLMs) on resource-constrained edge devices is increasingly hindered by prohibitive memory and computational requirements. While ternary quantization offers a compelling solution by reducing weights to $\{-1, 0, +1\}$, current implementations suffer from a fundamental misalignment with commodity hardware. Most existing methods must choose between 2-bit aligned packing, which incurs significant bit wastage, or 1.67-bit irregular packing, which degrades inference speed. To resolve this tension, we propose **Sherry**, a hardware-efficient ternary quantization framework. Sherry introduces a 3:4 fine-grained sparsity that achieves a regularized 1.25-bit width by packing blocks of four weights into five bits, restoring power-of-two alignment. Furthermore, we identify ***weight trapping*** issue in sparse ternary training, which leads to representational collapse. To address this, Sherry introduces **Arenas**, an annealing residual synapse mechanism that maintains representational diversity during training. Empirical evaluations on LLaMA-3.2 across five benchmarks demonstrate that Sherry matches state-of-the-art ternary performance while significantly reducing model size. Notably, on an Intel i7-14700HX CPU, our 1B model achieves zero accuracy loss compared to SOTA baselines while providing 25% bit savings and 10% speed up. The code is available at https://github.com/Tencent/AngelSlim.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across massive applications (Wu et al., 2023; Floridi and Chiriatti, 2020). However, the increasing concerns regarding data privacy, the necessity for offline functionality, the prohibitive latency, and the cost of cloud-based inference (Yao et al., 2024; Liagkou et al., 2024) have driven a pressing need to deploy LLMs on resource-constrained edge devices.

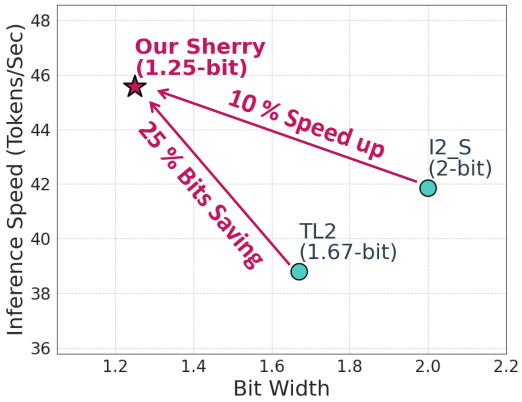

Figure 1: Comparison of different packing strategies for ternary quantization in efficiency.

Weight quantization (Hubara et al., 2018; Liu et al., 2023) is a practical technique for enabling on-device deployment by reducing model footprints and computation through lower-precision numerical representations. Nevertheless, most existing quantization methods (Dettmers et al., 2024; Lin et al., 2023b; Frantar et al., 2022) are optimized for server-grade GPUs that support complex hardware primitives, such as mixed-precision multiplication. Thus, these methods are often unsuitable for heterogeneous edge and mobile hardware, which prioritize simplified, widely supported operations that avoid specialized hardware dependencies.

Ternary quantization (Li et al., 2016; Liu and Liu, 2023) provides a compelling paradigm for edge-based LLM inference by constraining weights to the set $\{-1, 0, +1\}$. Supported by the Lookup Table (LUT)-based engines (Huang et al., 2025c; Wei et al., 2025; Wang et al., 2025a; Nie et al., 2025), ternary quantization transforms costly floating-point multiplications into hardware-efficient additions, as shown in Fig 9. This intrinsic compatibility makes ternary quantization a promising solution to bridge the gap between high-performance LLMs and the strict resource constraints of edge devices.

However, existing ternary quantization methods (Ma et al., 2025; Huang et al., 2025a) suf-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11186–11204

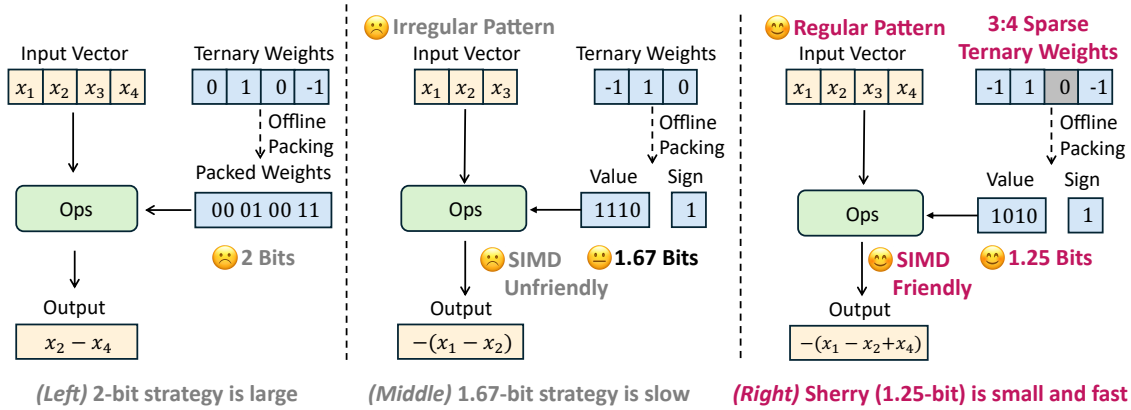

Figure 2: *(Left)* **2-bit strategy** packs each weight into 2 bits to maintain alignment, resulting in large bit wastage. *(Middle)* **1.67-bit strategy** packs 3 weights into 5 bits, introducing SIMD-unfriendly 3-way patterns, leading to slow speed. *(Right)* **Our Sherry** enforces a 3:4 sparsity and packs 4 weights into 5 bits, introducing SIMD-friendly 4-way patterns, achieving a small 1.25-bit width and faster inference speed. More details are shown in Fig 9.

fer from practical inefficiencies due to the misalignment between non-standard ternary bit-widths and standard hardware architectures. Current implementations generally resort to two suboptimal strategies for LUT-based execution: (1) **2-bit strategy** that packs each weight into 2 bits (Wei et al., 2025), as shown in Fig. 2 *(left)*, offering no memory savings over standard INT2 quantization; or (2) **1.67-bit strategy** that packs three weights into five bits (Wang et al., 2025a), as shown in Fig. 2 *(middle)*. While the 1.67-bit strategy offers improved density, its 3-way grouping is fundamentally incompatible with the power-of-two vector lanes of modern Single Instruction Multiple Data (SIMD) units, frequently leading to a slower inference speed compared to the 2-bit strategy. Consequently, **existing ternary methods are forced to trade off between bit width and inference speed**, preventing them from achieving the full theoretical benefits of ternary weights.

To address these limitations, we propose **Sherry**, a novel **S**parse **h**ardware-**e**fficient t**er**na**r**y quantization framework that achieves 1.25-bit width while maintaining superior inference speed. Our key insight is that the inherent sparsity of ternary models can be strategically structured to reconcile the tension between storage density and computational regularity. Specifically, Sherry enforces an optimal 3:4 fine-grained sparsity constraint: within every contiguous block of four weights, exactly three are quantized to non-zero values (± 1), and one is fixed to zero. This structured constraint enables an optimal packing strategy where each 4-weight block is stored in a compact 5-bit representation. Crucially, this block-based approach restores the power-of-two alignment required by modern SIMD units, allowing for parallel processing in regularized hardware operations.

Furthermore, directly integrating 3:4 sparsity into ternary training often results in performance degradation. We attribute this to ***weight trapping***: *weights accumulate in localized regions due to* ***gradient homogenization****, leading to representational collapse,* as shown in Fig. 3 and 11. To address this, Sherry introduces **Arenas**[1], an **A**nnealing **re**sidual sy**na**p**s**e module. By injecting heterogeneous gradients during the training phase, this module breaks the gradient homogenization and maintains an expressive and diverse weight distribution for Sherry.

We evaluate the efficacy and efficiency of Sherry across five standard benchmarks using the LLaMA-3.2 (Touvron et al., 2023) model family. Our empirical results demonstrate that Sherry matches performance with state-of-the-art (SOTA) ternary quantization baselines while necessitating a substantially lower bit width. Notably, on an Intel i7-14700HX CPU, Sherry achieves a 16% reduction in model size and a 10% inference speed up compared to SOTA 1B ternary LLM, with zero accuracy degradation. These findings validate that Sherry provides a promising hardware-efficient solution for deploying LLMs on resource-constrained platforms.

2 Background and Challenge

2.1 Ternary Quantization

Ternary quantization is an extreme weight compression paradigm that constrains model parameters to the discrete set $\{-1, 0, +1\}$. Taking per-channel quantization as an example, for a full-precision weight matrix $W \in \mathbb{R}^{d_{in} \times d_{out}}$, where d_{in} is the

[1] "Arenas" is also the name of the sandy soil used to grow grapes for sweet Sherry wine.

number of input channels and d_{out} is the number of output channels, the general ternary quantization function $Q(\cdot)$ is defined as:

$$Q(W) = T\alpha, \quad T_{i,j} = \begin{cases} +1, & \text{if } W_{i,j} > \Delta_j; \\ 0, & \text{if } |W_{i,j}| \le \Delta_j; \\ -1, & \text{if } W_{i,j} < -\Delta_j, \end{cases} \tag{1}$$

where $T \in \{-1, 0, +1\}^{d_{in} \times d_{out}}$ is the ternary weight matrix, $\alpha \in \mathbb{R}^{d_{out}}$ represents the scaling factors[2], and $\Delta \in \mathbb{R}^{d_{out}}$ denotes the quantization thresholds. A substantial body of research has investigated the optimal determination of α and Δ, which are discussed in detail in Sec. E.

To eliminate weight-activation multiplications during inference, **Lookup Table (LUT)-based engines** (Wang et al., 2025a; Wei et al., 2025) have been developed, demonstrating superior efficiency over traditional multiplication-based engines. As illustrated in Fig. 9, the engine segments the inputs and pre-computes a localized lookup table. For each segment, the corresponding ternary weight index is used to retrieve pre-computed results from the table. This paradigm provides a promising pathway for ternary LLMs by fully replacing complex floating-point multiplications with highly efficient additions and memory lookups.

2.2 Quantization-Aware Training (QAT)

Due to the aggressive nature of ternary compression, QAT is essential to recover model fidelity. In the forward pass, full-precision weights W are dynamically quantized via $Q(\cdot)$ in Eq. 20. During the backward pass, since $Q(\cdot)$ is non-differentiable, we employ the Straight-Through Estimator (STE) (Zhu et al., 2016; Chen et al., 2024b) to approximate gradients. For an input matrix $X \in \mathbb{R}^{d_t \times d_{in}}$, where d_t is the number of tokens, the forward pass and gradients for loss L are:

$$Y = XT\alpha, \quad \frac{\partial L}{\partial W} \approx X^\top \frac{\partial L}{\partial Y}. \tag{2}$$

After training, the weights W are discarded, leaving the ternary weights T and scaling factors α.

2.3 Challenge

Although ternary quantization has a theoretical lower bound of 1.58 bits ($\log_2 3$), contemporary implementations encounter significant architectural friction when deployed on commodity hardware. Most current methods resort to two suboptimal strategies: **(1) 2-bit Strategy:** This approach pads each ternary weight into 2 bits (Wei et al., 2025), as shown in Fig. 2 *(left)*. While it preserves computational regularity and aligns with SIMD vector lanes, it effectively nullifies the storage advantages of the ternary set compared to 2-bit integer quantization. **(2) 1.67-bit Strategy:** This scheme packs three weights into 5-bit blocks (Wang et al., 2025a), as shown in Fig. 2 *(middle)*. However, this introduces severe arithmetic inefficiencies because modern hardware accelerators are optimized for 2^n operand groupings. In practice, this misalignment often results in slower inference speeds than the 2-bit strategy.

Alternative strategies, such as packing 5 weights into 8 bits (1.6-bit), exponentially increase the size of the lookup tables, rendering them unsuitable for LUT-based edge engines. Consequently, existing ternary quantization remains trapped in a trade-off between memory footprint and inference speed, preventing it from realizing its full theoretical benefits in practical edge deployments.

3 Proposed Sherry

3.1 3:4 Sparse Ternary Quantization

The primary objective of Sherry is to achieve substantial bit savings without compromising inference throughput. Building on the observation that ternary models exhibit inherent sparsity (Liu and Liu, 2023), we formalize this characteristic by adopting an N:M structured sparsity constraint. This enforces a rigid pruning pattern where exactly N non-zero elements are permitted within every contiguous block of M weights.

While traditional $N : M$ sparsity (e.g., the 2:4 pattern) is typically coupled with specific GPU-vendor kernels (Zhou et al., 2021), Sherry leverages a multiplication-free LUT engine to decouple the sparsity constraint from specialized hardware primitives. This architectural freedom allows us to explore a 3:4 structured sparsity pattern, which achieves a highly efficient 1.25-bit width. The optimization goal of Sherry is to minimize the L_2 reconstruction error between the full-precision weights W and the sparse ternary representation $T\alpha$, subject to the 3:4 sparsity constraint. Let $W_{b:b+3,j} = (W_{b,j}, W_{b+1,j}, W_{b+2,j}, W_{b+3,j})$ denote a contiguous vector. The objective is formu-

[2] In this paper, we denote the multiplication between matrix T and vector α as element-wise, *i.e.*, $[T\alpha]_{i,j} = T_{i,j}\alpha_j$.

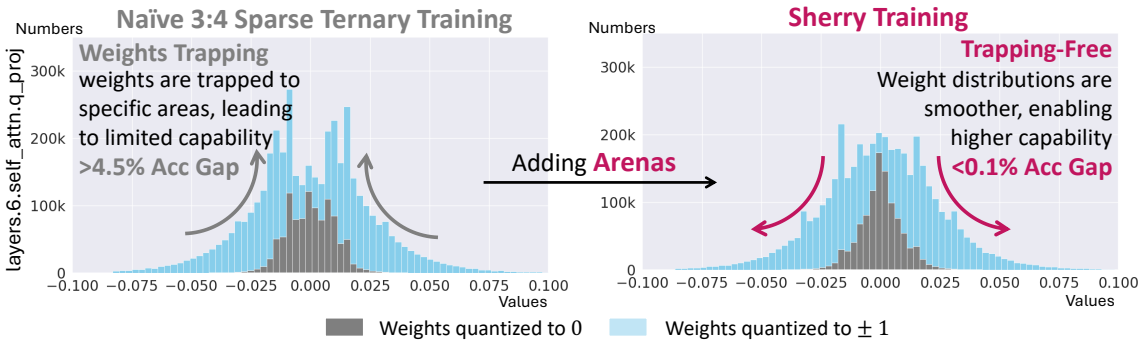

Figure 3: Comparison of weight distributions for LLaMA-1B under different quantization schemes. *(Left)* Naive 3:4 sparse ternary training exhibits **weight trapping**, where values collapse into a binary-like polarization, leading to a significant accuracy gap compared to dense models. *(Right)* Our Sherry utilizes the Arenas module to achieve a trap-free distribution, bridging the performance gap to dense ternary models.

lated as[3]:

$$\min_{T,\alpha} \sum_{j=1}^{d_{out}} \|W_{:,j} - T_{:,j}\alpha_j\|_2^2$$
$$\text{s.t. } T_{i,j} \in \{-1, 0, +1\},$$
$$\forall b \in \{1, 5, \ldots, d_{in} - 4\} : \|T_{b:b+3,j}\|_0 = 3, \quad (3)$$

where $\|\cdot\|_0 = 3$ enforces the 3:4 sparsity by ensuring exactly three non-zero values per block.

The optimal solution to the objective in Eq. 3 is obtained via a greedy Sparse-AbsMean strategy. For each block $W_{b:b+3,j}$, we prune the element with the smallest absolute magnitude and assign ternary values to the remaining three. Formally, for each block $i \in [b, b+3]$, the optimal ternary element $T_{i,j}$ is given by:

$$T^*_{i,j} = \begin{cases} 0, & \text{if } i = \arg\min |W_{i,j}|; \\ \text{sign}(W_{i,j}), & \text{otherwise.} \end{cases} \quad (4)$$

Given this optimal ternary matrix T, the optimal scaling factor α_j for the j-th output channel is calculated as the mean absolute value of the non-pruned weights:

$$\alpha^*_j = \frac{4}{3d_{in}} \sum_{i \in \mathcal{S}_j} |W_{i,j}|, \quad (5)$$

where $\mathcal{S}_j = \{i \mid T_{i,j} \neq 0\}$ denotes the set of active (non-zero) indices in the j-th column. The proof of the optimal is presented in Sec. D.

Although rarely used in floating-point contexts, 3:4 sparsity represents **an ideal "sweet spot"** for ternary quantization across four dimensions:

(1) SIMD-friendly Alignment: The choice of $M = 4$ ensures power-of-two alignment with activation segments, which is critical for SIMD vector lane loading. This eliminates the complex bit-shuffling overhead characteristic of 1.67-bit (3-way) packing schemes.

(2) Safe Sparsity Margin: Prior research (Zhu et al., 2016) indicates that ternary quantization performance undergoes severe degradation when the sparsity ratio exceeds 50%. 3:4 structured scheme maintains a 25% sparsity level within the safe margin required to preserve model expressive capacity.

(3) Optimal Bit-State Utilization: For $M = 4$, enforcing $N = 3$ non-zero elements yields exactly $\binom{4}{3} \times 2^3 = 32$ unique permutations. This mathematically saturates a 5-bit index ($2^5 = 32$), ensuring maximum bit-state utilization without bit wastage.

(4) Compatibility with LUT-Inference: The 3:4 pattern is natively compatible with the LUT-based ternary inference engine, as shown in Fig 9. Given the 128-bit constraints of standard SIMD register instructions (e.g., AVX2 vpshufb), the maximum capacity for a single-instruction lookup table is 16 bytes. By utilizing mirror-symmetry of ternary states (Ma et al., 2025; Wei et al., 2025), Sherry splits the 5-bit representation into 1 sign bit and 4 index bits, perfectly fitting within search limits.

In summary, the 3:4 format represents an intersection of hardware regularity and representational fidelity. **In Sec. C, We prove that the 3:4 format is optimal for LUT-based SIMD inference engines in terms of bit-state saturation and packing efficiency.**

3.2 Arenas: Annealing Residual Synapse

Despite the hardware and bit-utilization advantages of 3:4 structured sparsity, its direct application in ternary quantization often triggers significant performance degradation compared to the dense model. We identify the root cause as **weight trap-**

[3]In this paper, we denote $W_{:,j} \in \mathbb{R}^{d_{in}}$ as the j-th column o matrix $W \in \mathbb{R}^{d_{in} \times d_{out}}$, *i.e.*, $W_{:,j} = (W_{1,j}, \ldots W_{d_{in},j})$

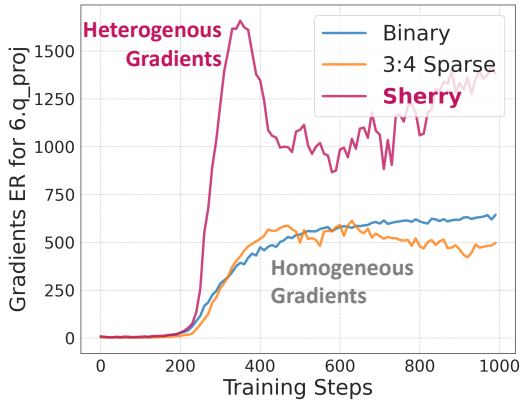

Figure 4: The effective ranks (ER) of the gradients during training. Both Binary and 3:4 sparse ternary training have relatively low ER due to gradient homogenization.

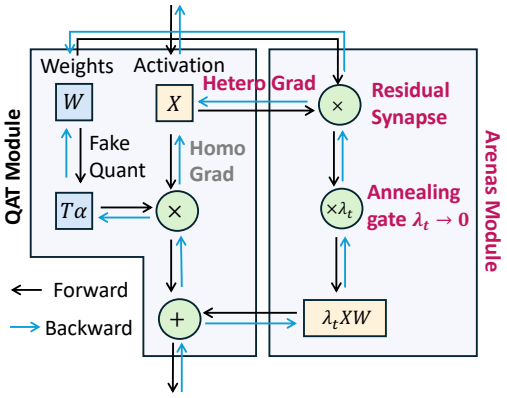

Figure 5: The overview of the Arenas module with QAT. The Arenas module injects the heterogeneous gradients through a residual synapse with an annealing gate.

ping. As illustrated in Fig. 3, under the hard 3:4 pruning constraint, weights tend to polarize toward specific values, resulting in a distribution that mimics binary quantization. This collapse prevents the model from exploiting the expressive capacity of the ternary set, essentially trapping it in a suboptimal binary-like state in Fig. 10 *(Top Right)*.

We find that this stagnation is driven by **Gradient Homogenization**. In a 3:4 sparsity configuration, the zeros are distributed uniformly within T, causing the sparse matrix to behave similarly to a dense binary matrix. This uniform distribution mimics the properties of a Hadamard transform (Huang et al., 2024b; Ashkboos et al., 2024), which flattens the signal. Consequently, the downstream gradients w.r.t. the activations X become increasingly homogenized:

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y}(T\alpha)^\top. \tag{6}$$

Therefore, the gradients passed to preceding layers become undifferentiated. This forces the majority of weights to exhibit similar behavior during training, significantly decreasing the representational diversity. In Fig. 4 *(left)*, the Effective Rank (ER) (Roy and Vetterli, 2007) for 3:4 sparse training confirms our analysis. Specifically, the 3:4 sparse regime exhibits a low ER ($ER < 750$), a level of spectral collapse comparable to that of binary quantization, despite the gradient matrix having a total dimensionality of 4096. This indicates a significant loss of learning degrees of freedom in naive 3:4 ternary sparse training (see Appendix F for more details).

To restore representational diversity, we propose Arenas, an Annealing Residual Synapse mechanism that re-couples latent weight magnitudes to the loss objective via a continuous bypass. During the training phase, the output of a ternary linear layer is augmented with a decaying full-precision residual synapse:

$$Y = XT\alpha + \lambda_t XW, \tag{7}$$

where λ_t is a scheduling coefficient that anneals to zero by the conclusion of training. The inclusion of the latent matrix W in the forward pass fundamentally alters the gradient dynamics. The gradient with respect to the latent activation X becomes:

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y}(T\alpha + \lambda_t W)^\top. \tag{8}$$

By incorporating the continuous values of W into the backward path, Arenas injects heterogeneous information back into $\frac{\partial L}{\partial X}$, breaking the homogenization effect induced by the 3:4 structure. This allows earlier layers to receive specialized updates, providing the "energy" required for weights to escape trapped binary-like states. As $\lambda_t \to 0$, the residual vanishes, leaving a pure 3:4 sparse ternary model for inference with zero additional overhead. The Arenas provides three critical advantages for 3:4 sparse ternary training:

(1) Variance Injection and Singularity Breaking: By re-introducing the continuous matrix W, Arenas prevents the gradients $\frac{\partial L}{\partial X}$ from collapsing into a homogenized, low-rank state, allowing earlier layers to receive specialized updates (in Fig. 4).

(2) Adaptive Error Compensation: During training, the residual term $\lambda_t XW$ naturally absorbs the quantization noise and pruning error introduced by the 3:4 ternary constraint. This allows the network to maintain a high-precision internal representation while the sparse ternary weights ($T\alpha$) gradually learn to capture the most salient components of the

Size	Method	Bit-width	ARC-e	ARC-c	HelS	PIQA	WinG	Average
1B	BF16	16	0.654	0.313	0.477	0.742	0.603	0.558
	LSQ	1.67	0.376	0.177	0.258	0.574	0.506	0.378
	SEQ	1.67	0.421	0.180	0.273	0.604	0.510	0.398
	DLT	1.67	0.424	0.174	0.256	0.563	0.513	0.386
	TWN	1.67	0.407	0.220	0.284	0.601	0.492	0.401
	AbsMedian	1.67	0.567	0.251	0.339	0.674	0.533	0.473
	AbsMean	1.67	0.603	0.259	0.360	0.683	0.541	**0.489**
	Tequila	1.67	**0.645**	**0.305**	**0.391**	**0.710**	**0.542**	**0.519**
	Sherry	**1.25**	**0.647**	**0.309**	**0.388**	**0.699**	**0.550**	**0.519**
3B	BF16	16	0.745	0.422	0.552	0.768	0.691	0.636
	LSQ	1.67	0.431	0.200	0.294	0.599	0.522	0.409
	SEQ	1.67	0.498	0.231	0.303	0.645	0.529	0.441
	DLT	1.67	0.361	0.161	0.260	0.572	0.496	0.370
	TWN	1.67	0.692	**0.351**	**0.462**	0.734	0.586	0.565
	AbsMedian	1.67	0.636	0.299	0.406	0.713	0.558	0.522
	AbsMean	1.67	0.672	0.329	0.439	0.735	0.582	0.551
	Tequila	1.67	**0.702**	0.346	**0.464**	**0.739**	**0.627**	**0.576**
	Sherry	**1.25**	**0.688**	**0.364**	0.452	**0.736**	**0.593**	**0.567**

Table 1: Comparison of Sherry with different ternary quantization methods.

signal, leading to better optimization.

(3) Zero-Overhead Inference: Because λ_t anneals to zero by the end of training, the auxiliary path is completely removed before deployment. The Arenas module therefore incurs zero inference-time overhead and requires no changes to the inference engine or weight format. The final deployed model is a standard 3:4 sparse ternary model identical to one trained without Arenas.

4 Evaluation

sherTo validate the efficacy of Sherry, we conduct experiments evaluating its performance against ternary quantization baselines. All results are averaged over three independent runs with random seeds. In all tables, the best and second-best results are highlighted in **purple** and **blue**; half-precision (BF16) results are shown in gray for reference.

4.1 Experimental Setup

We provide an overview of our experimental configuration below; further implementation details are available in Appendix G.

Datasets, Models, and Evaluation: We utilize LLaMA-3.2-1B and LLaMA-3.2-3B (Touvron et al., 2023) as our base architectures. Unless otherwise specified, we employ group-wise quantization with group size 128. For quantization-aware training, we utilize 10B tokens sampled from the UltraFineWeb dataset (Wang et al., 2025b). Following established ternary quantization benchmarks (Liu et al., 2025; Chen et al., 2024b; Ma et al., 2025), we evaluate performance using `lm-evaluation-harness` (Gao et al., 2024) across five zero-shot tasks: PIQA (Bisk et al., 2020), ARC-Easy (ARC-e), ARC-Challenge (ARC-c) (Clark et al., 2018), HellaSwag (HelS) (Zellers et al., 2019) and WinoGrande (WinG) (Sakaguchi et al., 2021). In our results, we report a bit-width of 1.67 for existing ternary quantization baselines rather than the theoretical 1.58, as 1.67 reflects the actual bit-density achieved by current practical packing strategies.

Ternary Quantization Baselines: We compare Sherry against two categories of quantization methods: (1) *Static methods*, including TWN (Li et al., 2016), Tequila used in TequilaLLM (Huang et al., 2025a), and the AbsMedian/AbsMean strategies utilized in BitNet (Wang et al., 2024; Nielsen and Schneider-Kamp, 2024) and Spectra (Kaushal et al., 2025); and (2) *Learnable methods*, such as DLT (Chen et al., 2024b) used in TernaryLLM, LSQ (Esser et al., 2019), and the SEQ strategy used in ParetoQ (Liu et al., 2025).

Implementation Details: Training is conducted on 32 NVIDIA GPUs, while inference throughput is evaluated on an Intel 8263C CPU to verify edge-efficiency. We quantize all linear layers within the Transformer architecture using a sequence length of 1024. The learning rate is fixed at 10^{-4}. The an-

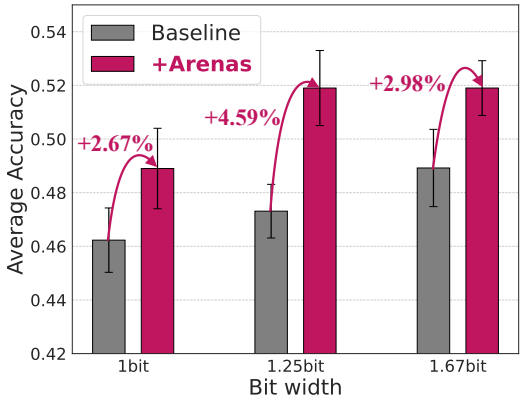

Figure 6: Ablation study of Arenas.

nealing coefficient λ_t follows a cosine-decay schedule with warmup by default, as shown in Fig. 7.

4.2 Performance Evaluation

Comparison of Ternary Quantization Methods: As shown in Table 1, **Sherry** achieves performance parity with the current SOTA, Tequila, across both 1B and 3B scales, despite utilizing a significantly lower bit-width of 1.25 bits. On the 1B model, Sherry matches the average SOTA accuracy exactly, effectively matching the 1.67-bit baseline while achieving a 25% reduction in bit-width. Notably, on reasoning-intensive benchmarks like ARC-Challenge, Sherry even outperforms Tequila and narrows the gap to the full-precision BF16 baseline to less than 0.5%. These results demonstrate that 3:4 structured sparsity preserves high-level linguistic capabilities, while the Arenas module successfully addresses optimization trapping. This synergy enables a high-efficiency packing strategy that simultaneously ensures superior hardware alignment and competitive model quality.

Comparison of Ternary LLMs: To further evaluate the effectiveness of Sherry, we denote the resulting models as **SherryLLM** and compare them against existing ternary LLaMA-based architectures. For a fair comparison, we reproduced methods with available training code (Liu et al., 2025; Chen et al., 2024b) on 10B tokens from the Ultra-FineWeb dataset using identical hyperparameters, while reporting original publication results for others. As shown in Table 2, **SherryLLM** achieves at least second-best average accuracy across both scales despite its significantly lower bit-width of 1.25 bits. These results confirm that the integration of 3:4 structured sparsity with the Arenas module preserves model expressivity, allowing SherryLLM to achieve a 25% bit-width reduction while maintaining competitive performance against baselines.

Impact of Quantization Granularity: Quantization granularity involves a critical trade-off between hardware efficiency and representational accuracy. While coarse-grained per-tensor quantization maximizes acceleration, it often introduces significant error, whereas fine-grained per-group strategies mitigate this at the cost of increased memory overhead for scaling factors. We evaluate Sherry across per-tensor, per-channel, and per-group (size 128) granularities on five benchmarks. As the results of average accuracy shown in Table 3, Sherry maintains robust performance across all levels with minimal degradation. This stability is primarily driven by the Arenas module, which provides a continuous gradient flow that allows latent weights to adapt to varying scaling constraints.

Effectiveness of Arenas: To formally demonstrate the effectiveness of our training framework, we conducted an ablation study across three quantization schemes: binary (1-bit), 3:4 structured sparse (1.25-bit), and pure ternary AbsMean (1.67-bit). As shown in the Fig 6, the integration of Arenas yields consistent performance improvements across all configurations. Notably, all the schemes exhibit significant gains. We attribute this to the Arenas's ability to mitigate the trapping issue in both 1-bit and 1.67-bit, as visualized in Fig. 10.

Inference Efficiency: To empirically validate the efficiency of Sherry, we evaluate the token generation speed against 1.67-bit (TL2) and 2-bit (I2_S) baselines within the BitNet.cpp framework (Wang et al., 2025a) using per-channel quantization, alongside a BF16 baseline. Experiments were conducted on an Intel i7-14700HX CPU using 700M and 3B BitNet variants. Both Sherry and the baselines utilize the BitNet.cpp paradigm. We conduct LUT-based efficiency experiments on x86 platforms because the ternary LUT inference baseline, TL2_0 in BitNet.cpp, relies on x86-specific SIMD instructions. The results in Table 4 demonstrate that Sherry outperforms 1.67-bit and 2-bit variants. Specifically, for the 3B model, Sherry achieves a 18% speedup over the 1.67-bit baseline. This improvement is attributed to our 3:4 structured sparsity and hardware-aligned 5-bit packing, which maximizes SIMD vector lane utilization and eliminates the bit-shuffling overhead inherent in non-power-of-two packing schemes.

Applicability to Multiplication-Based Engines: While Sherry is designed for LUT-based engines,

Model	Size	Bit-width	ARC-e	ARC-c	HelS	PIQA	WinG	Average
LLaMA3.2	1B	16	0.654	0.313	0.477	0.742	0.603	0.558
TernaryLLM*	1B	1.67	0.424	0.174	0.256	0.563	0.513	0.386
ParetoQ*	1B	1.67	0.421	0.180	0.273	0.604	0.510	0.398
LLM-QAT	1B	1.67	0.360	0.262	0.313	0.551	0.496	0.397
BitNet	1.3B	1.67	0.549	0.242	0.377	0.688	**0.558**	0.483
Spectra	1.1B	1.67	0.563	0.246	**0.388**	0.693	**0.555**	**0.489**
TequilaLLM	1B	1.67	**0.645**	**0.305**	**0.391**	**0.710**	0.542	**0.519**
SherryLLM	1B	**1.25**	**0.647**	**0.309**	**0.388**	**0.699**	0.550	**0.519**
LLaMA3.2	3B	16	0.745	0.422	0.552	0.768	0.691	0.636
TernaryLLM*	3B	1.67	0.361	0.161	0.260	0.572	0.496	0.370
ParetoQ*	3B	1.67	0.498	0.231	0.303	0.645	0.529	0.441
LLM-QAT	3B	1.67	0.445	0.307	0.434	0.627	0.506	0.464
BitNet	3B	1.67	0.614	0.283	0.429	0.715	0.593	0.527
Spectra	3.9B	1.67	0.660	0.319	**0.483**	**0.744**	**0.631**	**0.567**
TequilaLLM	3B	1.67	**0.702**	**0.346**	**0.464**	**0.739**	**0.627**	**0.576**
SherryLLM	3B	**1.25**	**0.688**	**0.364**	0.452	0.736	0.593	**0.567**

Table 2: Comparison of SherryLLM with LLaMA-based ternary LLMs across different model sizes; * indicates results obtained from our reproduction.

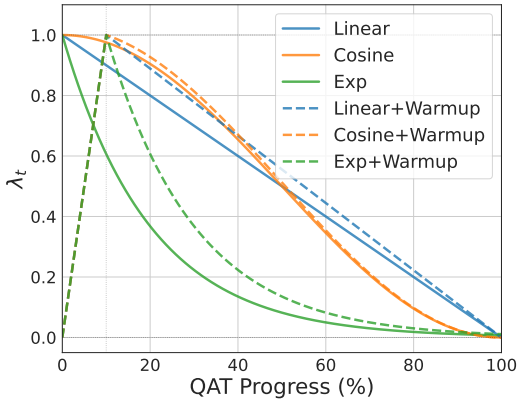

Figure 7: The schedules of annealing factor λ_t.

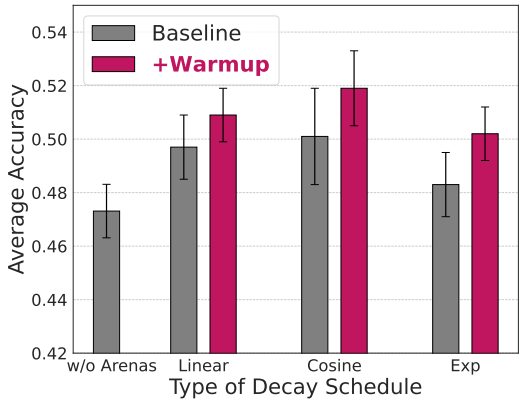

Figure 8: Comparison of different schedules for λ_t.

Granularity	Average Acc ± Std
Per-tensor	0.502 ± 0.010
Per-channel	0.513 ± 0.011
Per-group	0.519 ± 0.014

Table 3: Average accuracy of Sherry across various quantization granularities.

Scale	Method	Bits	Speed (t/s)↑	Size(MB)↓
0.7B	BF16	16	34.01	1360.0
	I2_S	2.0	**132.13**	256.56
	TL2	1.67	116.83	**233.44**
	Sherry	**1.25**	**148.27**	**205.50**
3B	BF16	16	7.55	6190.0
	I2_S	2.0	**41.87**	873.65
	TL2	1.67	38.80	**846.01**
	Sherry	**1.25**	**45.55**	**712.40**

Table 4: Inference efficiency on Intel i7-14700HX.

we also evaluate its behavior under a standard multiplication-based engine to assess generality. We benchmark Sherry against 1.67-bit (`TQ1_0`) and 2-bit (`TQ2_0`) baselines using `llama.cpp` on an Apple M4 Pro (ARM architecture). Since current hardware lacks native sub-8-bit arithmetic, all methods must dequantize weights to INT8 for computation; thus 2-bit packing represents an approximate upper bound for unpacking speed via simple bit shifts. As shown in Table 5, Sherry is faster than the 1.67-bit baseline and only marginally slower than the 2-bit baseline despite representing 37.5% fewer bits. This confirms that the 3:4 sparsity pattern is also efficient under multiplication-based engines on ARM, even though optimality guarantees are specific to the LUT-based inference context.

Training Overhead: Although the Arenas forward pass formally adds the term $\lambda_t XW$, it does not introduce an extra matrix multiplication. The output in Eq. 7 can be rewritten as $Y = X[Q(W) + W]$: a single matmul whose effective weight is the element-wise sum $Q(W) + W$. This reduces

Size	Quant	Bits	Speed (t/s) ↑
1B	FP16	16	70.66 ± 1.64
	TQ1_0	1.67	195.24 ± 7.21
	TQ2_0	2.0	239.14 ± 6.32
	Sherry	**1.25**	229.15 ± 6.26
3B	FP16	16	29.56 ± 0.93
	TQ1_0	1.67	77.95 ± 5.79
	TQ2_0	2.0	103.99 ± 5.72
	Sherry	**1.25**	95.33 ± 5.90

Table 5: Token generation speed on Apple M4 Pro with multiplication-based engine, llama.cpp.

the entire Arenas contribution to one element-wise weight addition, which can be fused directly into the fake-quantization step $Q(\cdot)$ with no additional GEMM calls. Empirically, this results in negligible overhead: per-step wall-clock time is 1.602s with Arenas versus 1.593s for naive QAT.

Impact of λ_t Schedules: To evaluate the sensitivity of the annealing process, we compare three decay schedules for the annealing gate λ_t: linear, exponential, and cosine, alongside their warmup counterparts. As shown in Fig. 8, every schedule consistently outperforms the baseline without Arenas, demonstrating the robust effectiveness of the Arenas regardless of the specific decay curve. Notably, warmup can increase performance for all schedules. We attribute it from early-stage optimization dynamics in warmup: by gradually introducing the residual, the model establishes a stable foundation before the annealing influence peaks.

5 Conclusion

In this work, we introduce Sherry, a novel ternary quantization framework designed to resolve the trade-off between bit width and inference speed. By leveraging a 3:4 structured sparsity pattern, Sherry achieves a 1.25-bit-width that natively aligns with SIMD vector lanes, effectively resolving the hardware under-utilization issues common in previous ternary packing strategies. To combat the performance degradation due to weight trapping, we proposed the Arenas module. Arenas provides an annealing, heterogeneous gradient flow during the training phase, allowing sparse models to retain the expressive diversity. Our extensive evaluations on the LLaMA-3.2 models demonstrate that Sherry achieves performance parity with SOTA ternary models while utilizing 25% fewer bits and 10% speed up. Future work will explore composing Sherry with activation and KV-cache quantization to further reduce memory pressure during long-context inference on edge devices.

Acknowledgement

This paper is partially supported by Hong Kong Research Grants Council grant C1042-23GF and grant #11203523.

Limitations

While Sherry advances the efficiency of edge-deployed LLMs, several limitations offer avenues for future research:

Edge-Centric Model Scale: Our evaluation focuses on models up to 3B parameters, as these are the primary candidates for local edge deployment. While we demonstrate that Sherry achieves a superior Pareto frontier for these scales, the behavior of the Arenas mechanism and the 3:4 sparsity pattern on larger, server-grade models (70B+) remains to be validated.

Lack of Server-Specific Optimizations: To prioritize edge inference efficiency, our experiments focus on general SIMD-aligned vector packing rather than server-specific hardware optimizations, such as NVIDIA Sparse Tensor Cores. While this does not affect Sherry's utility for mobile and local processors, future benchmarks on data-center GPUs could further explore its potential for high-throughput server applications.

Limited Platform Baselines: Our LUT-based inference comparisons are currently limited to x86 architectures, as the reference implementations for the 1.67-bit LUT baseline (`TL2_0`) depend on x86-specific instructions. Porting Sherry and the required LUT baselines to ARM and mobile processors is a prioritized direction for future work, with preliminary results on a multiplication-based ARM engine already demonstrating competitive throughput.

Weight-Only Quantization: To maximize weight-streaming throughput on edge devices, this study focuses on weight-only ternary quantization. While 1.25-bit weights drastically reduce the static memory footprint, activations and the KV cache remain in BF16. Future integration with activation and KV-cache quantization could further alleviate memory bottlenecks during long-context inference; we discuss composability with such methods in Appendix I.

References

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. 2025. Efficientqat: Efficient quantization-aware training for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10081–10100.

Ning Chen, Tie Qiu, Xiaobo Zhou, Songwei Zhang, Weisheng Si, and Dapeng Oliver Wu. 2024a. A distributed co-evolutionary optimization method with motif for large-scale iot robustness. *IEEE/ACM Transactions on Networking*.

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

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

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2021. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

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

Luciano Floridi and Massimo Chiriatti. 2020. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694.

George H. Forman and John Zahorjan. 1994. The challenges of mobile computing. *Computer*, 27(4):38–47.

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

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2023. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 12799–12807.

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

Hong Huang and Dapeng Wu. 2025. Quaff: Quantized parameter-efficient fine-tuning under outlier spatial stability hypothesis. *arXiv preprint arXiv:2505.14742*.

Hong Huang, Decheng Wu, Rui Cen, Guanghua Yu, Zonghang Li, Kai Liu, Jianchen Zhu, Peng Chen, Xue Liu, and Dapeng Wu. 2025a. Tequila: Trapping-free ternary quantization for large language models. *arXiv preprint arXiv:2509.23809*.

Hong Huang, Hai Yang, Yuan Chen, Jiaxun Ye, and Dapeng Wu. 2025b. Fedrts: Federated robust pruning via combinatorial thompson sampling. *arXiv preprint arXiv:2501.19122*.

Hong Huang, Lan Zhang, Chaoyue Sun, Ruogu Fang, Xiaoyong Yuan, and Dapeng Wu. 2023. Distributed pruning towards tiny neural networks in federated learning. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pages 190–201. IEEE.

Hong Huang, Weiming Zhuang, Chen Chen, and Lingjuan Lyu. 2024a. Fedmef: Towards memory-efficient federated dynamic pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27548–27557.

Xijie Huang, Zechun Liu, Shih-Yang Liu, and Kwang-Ting Cheng. 2024b. Rolora: Fine-tuning rotated outlier-free llms for effective weight-activation quantization. *arXiv preprint arXiv:2407.08044*.

Zhirui Huang, Rui Ma, Shijie Cao, Ran Shu, Ian Wang, Ting Cao, Chixiao Chen, and Yongqiang Xiong. 2025c. Tenet: An efficient sparsity-aware lut-centric architecture for ternary llm inference on edge. *arXiv preprint arXiv:2509.13765*.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2018. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30.

Tomasz Imielinski and Henry F Korth. 1996. *Mobile computing*, volume 353. Springer Science & Business Media.

Ayush Kaushal, Tejas Vaidhya, Arnab Kumar Mondal, Tejas Pandey, Aaryan Bhagat, and Irina Rish. 2025. Surprising effectiveness of pretraining ternary language model at scale. In *The Thirteenth International Conference on Learning Representations*.

Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. 2018. Extremely low bit neural network: Squeeze the last bit out with admm. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. 2016. Ternary weight networks. *arXiv preprint arXiv:1605.04711*.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. 2023. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*.

Zonghang Li, Tao Li, Wenjiao Feng, Mohsen Guizani, and Hongfang Yu. 2025. Prima. cpp: Speeding up 70b-scale llm inference on low-resource everyday home clusters. *arXiv preprint arXiv:2504.08791*.

Vasiliki Liagkou, Evangelia Filiopoulou, George Fragiadakis, Mara Nikolaidou, and Christos Michalakelis. 2024. The cost perspective of adopting large language model-as-a-service. In *2024 IEEE International Conference on Joint Cloud Computing (JCC)*, pages 80–83. IEEE.

Bin Lin, Ningxin Zheng, Lei Wang, Shijie Cao, Lingxiao Ma, Quanlu Zhang, Yi Zhu, Ting Cao, Jilong Xue, Yuqing Yang, and Fan Yang. 2023a. Efficient gpu kernels for n:m-sparse weights in deep learning. In *Proceedings of Machine Learning and Systems*, volume 5, pages 513–525. Curan.

Haokun Lin, Haobo Xu, Yichen Wu, Ziyu Guo, Renrui Zhang, Zhichao Lu, Ying Wei, Qingfu Zhang, and Zhenan Sun. 2025. Quantization meets dllms: A systematic study of post-training quantization for diffusion llms. *arXiv preprint arXiv:2508.14896*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023b. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Dan Liu and Xue Liu. 2023. Ternary quantization: A survey. *arXiv preprint arXiv:2303.01505*.

Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. 2023. Qllm: Accurate and efficient low-bitwidth quantization for large language models. *arXiv preprint arXiv:2310.08041*.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, Lin Xiao, Yuandong Tian, Bilge Soran, Raghuraman Krishnamoorthi, Tijmen Blankevoort, and Vikas Chandra. 2025. Paretoq: Improving scaling laws in extremely low-bit llm quantization. *Preprint*, arXiv:2502.02631.

Shuming Ma, Hongyu Wang, Shaohan Huang, Xingxing Zhang, Ying Hu, Ting Song, Yan Xia, and Furu Wei. 2025. Bitnet b1. 58 2b4t technical report. *arXiv preprint arXiv:2504.12285*.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Xin Nie, Liang Dong, Haicheng Zhang, Jiawang Xiao, and G Sun. 2025. Elutq: Efficient lut-aware quantization for deploying large language models on edge devices. *arXiv preprint arXiv:2510.19482*.

Jacob Nielsen and Peter Schneider-Kamp. 2024. Bitnet b1. 58 reloaded: State-of-the-art performance also on smaller networks. In *International Conference on Deep Learning Theory and Applications*, pages 301–315. Springer.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2023. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*.

Olivier Roy and Martin Vetterli. 2007. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE.

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

Wei Sun, Aojun Zhou, Sander Stuijk, Rob Wijnhoven, Andrew Oakleigh Nelson, hongsheng Li, and Henk Corporaal. 2021. Dominosearch: Find layer-wise fine-grained n:m sparse schemes from dense neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 20721–20732. Curran Associates, Inc.

MiniCPM Team, Chaojun Xiao, Yuxuan Li, Xu Han, Yuzhuo Bai, Jie Cai, Haotian Chen, Wentong Chen, Xin Cong, Ganqu Cui, Ning Ding, Shengda Fan, Yewei Fang, Zixuan Fu, Wenyu Guan, Yitong Guan, Junshao Guo, Yufeng Han, Bingxiang He, and 64 others. 2025. Minicpm4: Ultra-efficient llms on end devices. *Preprint*, arXiv:2506.07900.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton

Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Jinheng Wang, Hansong Zhou, Ting Song, Shijie Cao, Yan Xia, Ting Cao, Jianyu Wei, Shuming Ma, Hongyu Wang, and Furu Wei. 2025a. Bitnet. cpp: Efficient edge inference for ternary llms. *arXiv preprint arXiv:2502.11880*.

Jinheng Wang, Hansong Zhou, Ting Song, Shaoguang Mao, Shuming Ma, Hongyu Wang, Yan Xia, and Furu Wei. 2024. 1-bit ai infra: Part 1.1, fast and lossless bitnet b1. 58 inference on cpus. *arXiv preprint arXiv:2410.16144*.

Yudong Wang, Zixuan Fu, Jie Cai, Peijun Tang, Hongya Lyu, Yewei Fang, Zhi Zheng, Jie Zhou, Guoyang Zeng, Chaojun Xiao, Xu Han, and Zhiyuan Liu. 2025b. Ultra-fineweb: Efficient data filtering and verification for high-quality llm training data. *Preprint*, arXiv:2505.05427.

Jianyu Wei, Shijie Cao, Ting Cao, Lingxiao Ma, Lei Wang, Yanyong Zhang, and Mao Yang. 2025. T-mac: Cpu renaissance via table lookup for low-bit llm deployment on edge. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 278–292.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. 2022. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35:17402–17414.

Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. 2023. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136.

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

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

Lianwei Yang, Haokun Lin, Tianchen Zhao, Yichen Wu, Hongyu Zhu, Ruiqi Xie, Zhenan Sun, Yu Wang, and Qingyi Gu. 2025. Lrq-dit: Log-rotation post-training quantization of diffusion transformers for image and video generation. *arXiv preprint arXiv:2508.03485*.

Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2):100211.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Yunyun Sun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. 2023. Dynamic sparse no training: Training-free fine-tuning for sparse llms. *arXiv preprint arXiv:2310.08915*.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. 2021. Learning n: m fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010*.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. 2016. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*.

Appendix

A Inference Engine Design

Sherry seamlessly integrates with the BitNet.cpp (Wang et al., 2025a) framework to enable efficient, multiplication-free inference. As illustrated in Fig. 9, our engine operates in two distinct phases: an offline packing stage and an efficient online inference stage.

Offline Packing Phase During the offline phase, the 3:4 sparse ternary weights are compressed into a hardware-aligned format consisting of *index* and *sign* components. To maximize SIMD efficiency and memory density, every four-element block is packed into a 5-bit metadata structure: a 4-bit index represents the magnitude/sparsity pattern within the block, and a 1-bit value represents the shared or dominant sign. This packing strategy ensures that the weights are aligned with standard word boundaries, significantly reducing the bit-shuffling overhead compared to non-power-of-two schemes.

Online Inference Phase At inference time, the engine utilizes a high-performance lookup table (LUT) paradigm. The input activations are preprocessed into local LUTs for each weight segment. For each segment, the corresponding weight index serves as a direct pointer to retrieve precomputed results from the LUT, entirely replacing floating-point multiplications with memory-efficient lookups.

Following retrieval, the sign weight is applied to determine the final polarity of the segment sum. Subsequent accumulation across segments is performed via integer addition, concluding with the addition of the channel-wise scaling factor α. This architecture results in a highly optimized inference path that realizes the theoretical efficiency of 1.25-bit quantization while requiring only minimal modifications to existing low-bit inference kernels.

B Related Work

B.1 Quantization for LLMs

Quantization has emerged as a cornerstone technique for enhancing the efficiency of Large Language Models (LLMs) by reducing the bit-precision of weights and activations (Dettmers et al., 2021, 2022; Lin et al., 2023b; Frantar et al., 2022; Lin et al., 2025; Yang et al., 2025; Li et al., 2023). While effective, popular weight-only quantization methods (Lin et al., 2023b; Frantar et al., 2022) typically necessitate mixed-precision matrix multiplication, where weights and activations reside in disparate data formats. This heterogeneity requires specialized hardware support to maintain throughput, posing a significant hurdle for deployment on diverse edge and mobile platforms where hardware specialized for non-standard bit-widths is often unavailable.

To address this, weight-activation quantization strategies (Dettmers et al., 2022; Xiao et al., 2023; Huang and Wu, 2025; Wei et al., 2022) seek a unified low-precision format. However, these methods often encounter the "outlier problem" (Xiao et al., 2023; Dettmers et al., 2022), where extreme activation values lead to high quantization errors. Consequently, it remains challenging for activations to reach the same ultra-low precision as weights without severe performance degradation, leaving a gap in achieving truly hardware-agnostic, high-efficiency deployment on the edge.

B.2 Ternary Quantization

Ternary quantization (Li et al., 2016; Zhu et al., 2016), often referred to as 1.58-bit quantization, offers a paradigm shift by constraining weights to the set $\{-1, 0, +1\}$. Beyond substantial memory reduction, this approach fundamentally simplifies the core matrix multiplication operation into addition, effectively eliminating the need for power-intensive multipliers (Ma et al., 2025; Wang et al., 2024). This intrinsic hardware-friendliness is particularly vital for resource-constrained mobile environments.

Early research in this domain primarily focused on optimizing the quantization function through thresholding and scaling. Ternary Weight Networks (TWN) (Li et al., 2016) minimized reconstruction distortion by assuming a Gaussian weight distribution, while Trained Ternary Quantization (TTQ) (Zhu et al., 2016) introduced learnable scaling factors to adapt the ternary representation during training. Subsequent work by Leng et al. (2018) leveraged the Alternating Direction Method of Multipliers (ADMM) to iteratively optimize these parameters.

The advent of Large Language Models (LLMs) (Wu et al., 2023; Floridi and Chiriatti, 2020; Zhang et al., 2022) has revitalized interest in ternary schemes, as the generative capacity of LLMs is notoriously sensitive to precision loss. This has led to two distinct research trajectories. The first utilizes Post-Training Quantiza-

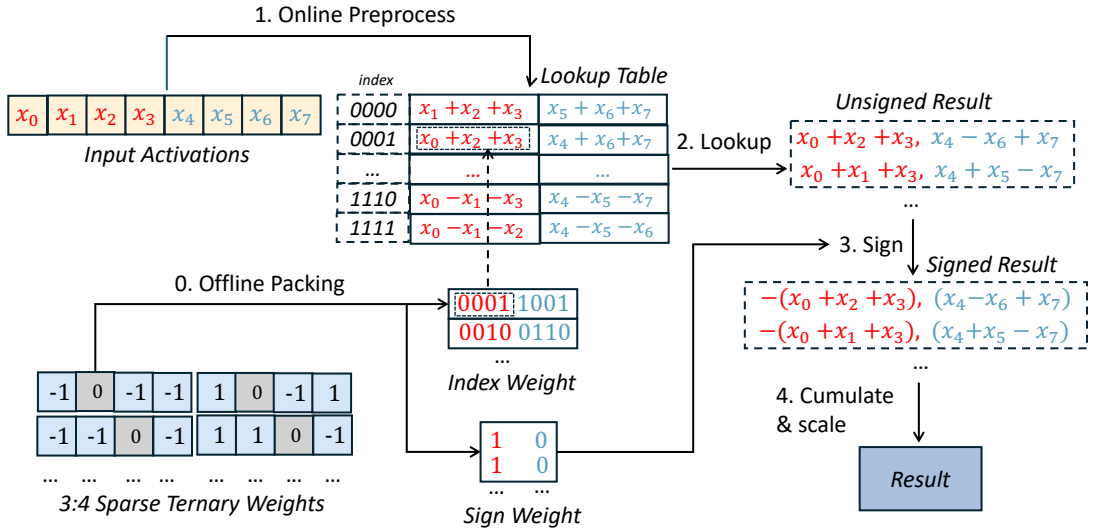

Figure 9: The LUT-Based Inference engine for **Sherry**. The inference engine constructs a dynamic lookup table from input activations; weight indices are then used to fetch pre-computed values.

tion (PTQ) (Lin et al., 2023b; Frantar et al., 2022; Xiao et al., 2025) to minimize the computational cost of quantization; however, PTQ often suffers from non-negligible performance drops at ultra-low bit-widths. Consequently, Quantization-Aware Training (QAT) (Chen et al., 2025) has become the preferred choice for robust performance recovery. Within the QAT landscape, strategies range from the straightforward AbsMean scaling utilized by the BitNet family (Ma et al., 2025; Wang et al., 2024, 2025a) to more sophisticated adaptations of Learned Step Size Quantization (LSQ) (Esser et al., 2019) for the ternary regime (Chen et al., 2024b; Liu et al., 2025).

Regarding the inference engine, classical frameworks like `llama.cpp` still rely on multiplication-based kernels that require weight unpacking and floating-point operations. To overcome this, T-Mac (Wei et al., 2025) and TENET (Huang et al., 2025c) proposed a lookup table (LUT)-based engine to eliminate multiplications, though it requires packing ternary weights into 2-bit containers. BitNet.cpp (Wang et al., 2025a) attempted to reduce the footprint further via a 1.67-bit packing strategy (3 weights in 5 bits). However, this 3-way packing is inherently SIMD-unfriendly, introducing significant bit-shuffling overhead that often results in slower performance than 2-bit packing.

Consequently, existing ternary models are forced to trade off between bit width and inference speed. Sherry addresses this dilemma by introducing hardware-aligned 3:4 structured sparsity, achieving a 1.25-bit SIMD-friendly packing.

B.3 N:M sparsity

N:M structured sparsity (Lin et al., 2023a; Zhou et al., 2021) has emerged as a critical middle ground between the flexibility of unstructured pruning and the hardware efficiency of block-wise pruning. By enforcing that exactly N out of every M contiguous weights are non-zero, this pattern provides predictable memory access patterns highly amenable to hardware acceleration. Notably, NVIDIA's Ampere and subsequent architectures (Lin et al., 2023a) introduced native Tensor Core support for 2:4 sparsity, doubling throughput with minimal accuracy loss in high-precision models.

Existing research (Sun et al., 2021; Fu et al., 2023; Zhang et al., 2023) on N:M sparsity has primarily focused on the selection of optimal masks and performance recovery via specialized training regimes. However, most efforts are not coordinated with ultra-low bit quantization, as they are largely designed for Sparse Tensor Cores on GPUs, which currently prioritize 16-bit or 32-bit floating-point arithmetic.

In the context of ultra-low bit-widths, the intersection of N:M sparsity and ternary quantization remains largely unexplored. Furthermore, as the bit-width decreases, the rigid N:M constraint exacerbates the **weight trapping** phenomenon. Sherry extends the N:M paradigm by introducing a hardware-aligned 3:4 pattern coupled with the Arenas module. This synergy is specifically designed to bridge the accuracy gap between sparse and dense ternary models with zero additional inference overhead. *To our knowledge, Sherry is the first to provide a*

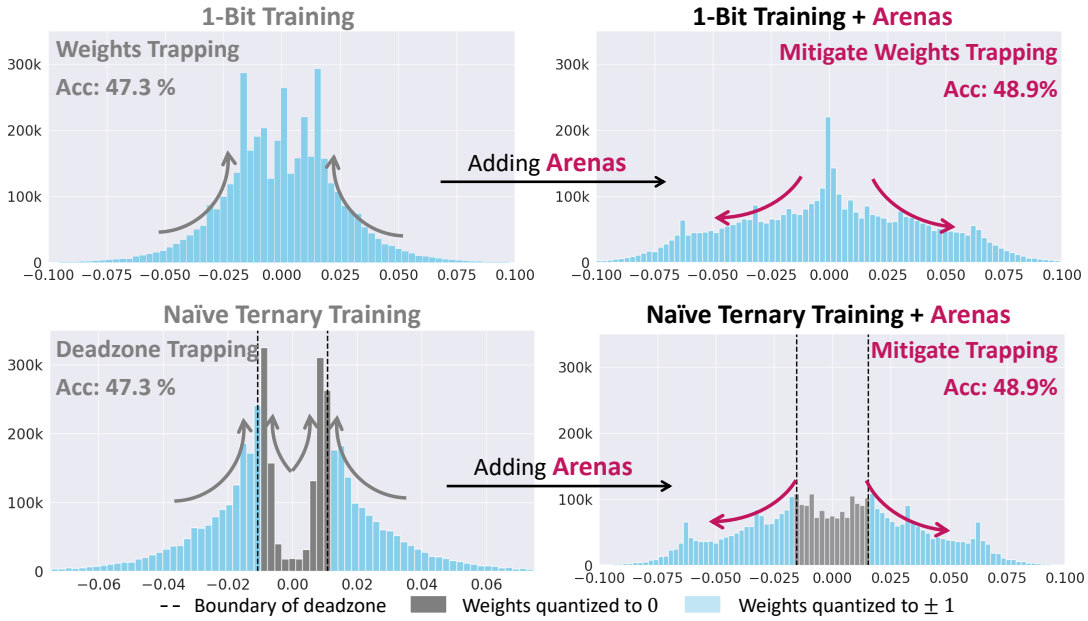

Figure 10: Weight distributions across different quantization regimes. The inclusion of Arenas effectively mitigates the trapping phenomenon in both binary and naive ternary quantization.

hardware-efficient solution for applying N:M sparsity on ultra-low bit-width quantization.

C Optimality of the 3:4 Sparse Format

We argue that the 3:4 structured sparsity pattern represents the optimal solution among all $N:M$ ($N < M$) formats for LUT-based SIMD ternary inference engines. To formalize this, we define the search space as packing M ternary weights into B bits, where B/M determines the effective bit-width. In a hardware-aligned implementation, these B bits are partitioned into 1 sign bit and $B-1$ index bits (utilizing the mirror-symmetry of ternary states).

C.1 Design Constraints

The selection of N and M is governed by three critical hardware and performance constraints:

(1) **SIMD Alignment:** The block size M must be a power of two ($M \in \{2^k\}$) to ensure predictable memory access and alignment with standard SIMD register widths.

(2) **LUT Capacity:** Standard SIMD instructions (e.g., x86 AVX2 vpshufb) utilize a 128-bit (16-byte) register as a lookup table. This constrains the index to 4 bits ($2^4 = 16$ entries), implying $B - 1 \leq 4$.

(3) **Sparsity Threshold:** To preserve representational capacity and avoid severe accuracy degradation (Zhu et al., 2016), the density ratio must satisfy $\frac{N}{M} \geq 0.5$ (i.e., sparsity $\leq 50\%$).

C.2 Proof of Optimality

By applying hardware Constraints (1) and (2), the search space for M is limited to $\{2, 4, 8\}$. For a target bit-width of approximately 1.25 bits, we evaluate the following candidate configurations:

- **M=2 (1:2):** Requires $B = 4$ bits ($B/M = 2.0$), failing the efficiency requirement.
- **M=8 (5:8, 6:8, 7:8):** These require $B \geq 10$ bits to maintain efficiency, leading to $B - 1 \geq 9$. This violates Condition (2), the 4-bit LUT capacity of a single SIMD instruction, requiring multi-cycle split-table lookups.
- **M=4 (2:4, 3:4):** These are the only candidates satisfying both hardware constraints.

To choose between 2:4 and 3:4, we consider the information density of the index space. A 3:4 ternary block with one shared sign bit has a total of $C_4^3 \cdot 2^{3-1} = 16$ unique patterns. This perfectly saturates the 2^4 entries in the LUT. In contrast, a 2:4 scheme only utilizes $C_4^2 \cdot 2^{2-1} = 12$ states, resulting in bit-waste.

Furthermore, 2:4 sparsity resides exactly on the 50% threshold where performance begins to destabilize. The 3:4 format provides 75% density (25% sparsity), staying well within the safe margin for model expressive capacity while maximizing bit-utilization. Thus, the 3:4 format is the optimal

solution for high-performance 1.25-bit SIMD inference.

D Optimality of the Sparse-Absmean

Let $W_{b:b+3,j} = (W_{b,j}, W_{b+1,j}, W_{b+2,j}, W_{b+3,j})$ denote a contiguous vector. The objective is formulated as:

$$\begin{aligned} &\min_{T,\alpha} \sum_{j=1}^{d_{out}} \|W_{:,j} - T_{:,j}\alpha_j\|_2^2 \\ &\text{s.t. } T_{i,j} \in \{-1, 0, +1\}, \\ &\quad \forall b \in \{1, 5, \ldots, d_{in} - 4\} : \|T_{b:b+3,j}\|_0 = 3. \end{aligned} \tag{9}$$

where $\|\cdot\|_0 = 3$ enforces the 3:4 structured sparsity by ensuring exactly three non-zero ternary values per block.

To determine the optimal indices and signs for T, we expand the per-block objective for a block b:

$$\min_{T,\|T\|_0=3} \sum_{i=b}^{b+3} (W_{i,j}^2 - 2\alpha_j W_{i,j} T_{i,j} + \alpha_j^2 T_{i,j}^2). \tag{10}$$

Since $\sum T_{i,j}^2 = 3$ is constant for any valid 3:4 pattern, minimizing the error is equivalent to maximizing the correlation term:

$$\max_{T,\|T\|_0=3} \sum_{i=b}^{b+3} W_{i,j} T_{i,j}. \tag{11}$$

To maximize this sum, we must choose $T_{i,j} = \text{sign}(W_{i,j})$ for the three elements with the largest absolute magnitudes and set $T_{i,j} = 0$ for the element with the smallest absolute magnitude in the block.

Sign Optimality: For any element i where $T_{i,j} \neq 0$, the quadratic term $(W_{i,j} - T_{i,j}\alpha)^2$ is minimized when $T_{i,j}$ has the same sign as $W_{i,j}$. If $\text{sign}(T_{i,j}) \neq \text{sign}(W_{i,j})$, flipping the sign of $T_{i,j}$ would increase the correlation $W_{i,j}T_{i,j}$ and strictly decrease the objective for any $\alpha > 0$.

Index Optimality: Let $\{|W|_{(1)}, |W|_{(2)}, |W|_{(3)}, |W|_{(4)}\}$ be the sorted absolute values of weights in a 4-element block such that $|W|_{(1)} \geq |W|_{(2)} \geq |W|_{(3)} \geq |W|_{(4)}$. The correlation sum for the block is $\sum_{k\in\mathcal{I}} |W|_{(k)}$, where $\mathcal{I}$ is the set of indices of the 3 chosen elements. To maximize this sum, we must choose the indices corresponding to the three largest magnitudes: (1), (2), and (3).

By combining the sign and index selection, for each block $i \in [b, b+3]$, the optimal ternary element $T_{i,j}$ is given by:

$$T_{i,j}^* = \begin{cases} 0, & \text{if } i = \arg\min |W_{i,j}|; \\ \text{sign}(W_{i,j}), & \text{otherwise.} \end{cases} \tag{12}$$

Substituting $T_{i,j}^*$ back into the expression for α_j^* gives the global minimum:

$$\alpha_j^* = \frac{4}{3d_{in}} \sum_{i\in\mathcal{S}_j} |W_{i,j}|, \tag{13}$$

where $\mathcal{S}_j = \{i \mid T_{i,j} \neq 0\}$ denotes the set of active (non-zero) indices in the j-th column.

E Ternary Quantization Baselines

Recall the general form of ternary quantization: taking per-channel quantization as an example, for a full-precision weight matrix $W \in \mathbb{R}^{d_{in}\times d_{out}}$, the general ternary quantization function $Q(\cdot)$ is defined as:

$$Q(W) = T\alpha, \quad T_{i,j} = \begin{cases} +1, & \text{if } W_{i,j} > \Delta_j; \\ 0, & \text{if } |W_{i,j}| \leq \Delta_j; \\ -1, & \text{if } W_{i,j} < -\Delta_j, \end{cases} \tag{14}$$

where $T \in \{-1, 0, +1\}^{d_{in}\times d_{out}}$ is the ternary weight matrix, $\alpha \in \mathbb{R}^{d_{out}}$ represents the scaling factors, and $\Delta \in \mathbb{R}^{d_{out}}$ denotes the quantization thresholds.

For contemporary large-scale ternary models (Ma et al., 2025; Team et al., 2025), the AbsMean method has emerged as the de facto standard due to its superior training stability. In AbsMean, the scaling factor α_j and threshold Δ_j for the j-th channel are calculated as:

$$\alpha_j = \frac{1}{d_{in}} \sum_{i=1}^{d_{in}} |W_{i,j}|, \quad \Delta_j = \frac{\alpha_j}{2}. \tag{15}$$

Besides Absmean, previous optimization strategies can be broadly categorized into: (1) reducing quantization error and (2) enhancing model expressive capacity.

E.1 Reducing Quantization Error

A primary line of work focuses on optimizing Δ and α to minimize the reconstruction error

$$\min_{\Delta,\alpha} \|W - T\alpha\|_2^2. \tag{16}$$

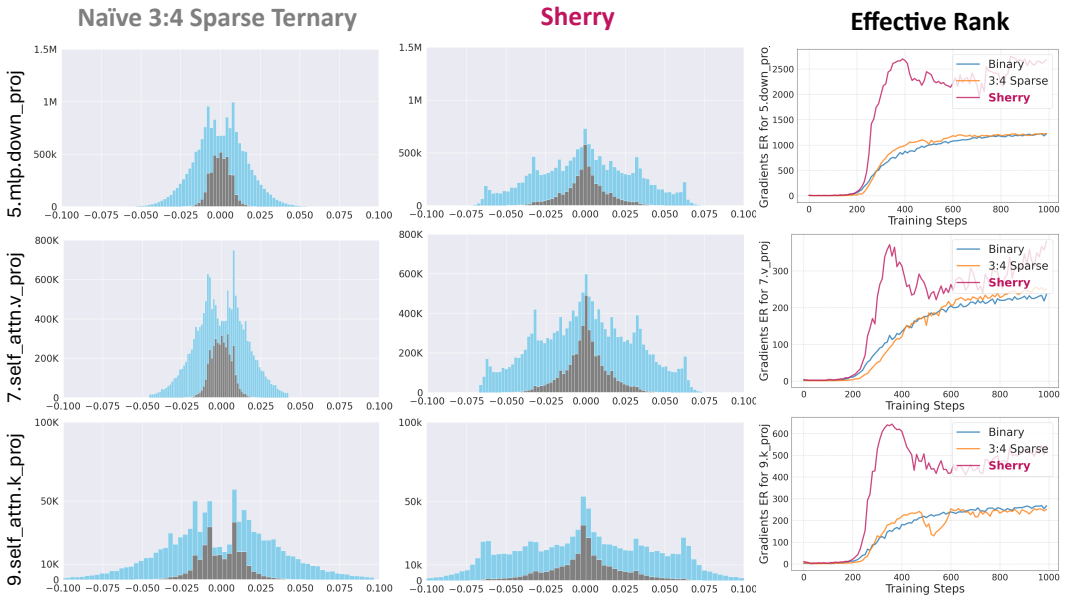

Figure 11: Weight distributions and the Effective Ranks (ER) of their gradients across different layers.

This is exemplified by estimation-based methods like Ternary Weight Networks (TWN) (Li et al., 2016), which minimize distortion by assuming a Gaussian weight distribution to approximate

$$\Delta_j^* \approx 0.7 \cdot \mathbb{E}[|W_{:,j}|]. \tag{17}$$

For a fixed Δ, the optimal α is:

$$\alpha_j^* = \frac{1}{|\mathcal{S}_j|} \sum_{i \in \mathcal{S}_j} |W_{i,j}|, \tag{18}$$

where $\mathcal{S}_j$ is the set of weights $W_{i,j}$ exceeding Δ_j. However, the Gaussian assumption often fails in LLMs, leading to biased estimates and performance degradation. While subsequent methods like LSQ (Esser et al., 2019) and DLT (Chen et al., 2024b) introduce trainable scaling factors, they are easier to being trapped in sub-optima, exhibiting slower convergence and higher final loss compared to AbsMean (Huang et al., 2025a).

E.2 Enhancing Expressive Capacity

Alternative approaches attempt to recover capacity by incorporating bias terms. DLT (Chen et al., 2024b) introduces a learnable bias during dequantization:

$$Y = X(T\alpha + B) = XT\alpha + Xb. \tag{19}$$

However, this necessitates a dense full-precision multiplication (Xb), destroying the computational efficiency of ternary quantization. Similarly, SEQ (Liu et al., 2025) reassigns the zero-point to a non-zero value $\alpha_j b_j$:

$$T_{i,j} = \begin{cases} +1, & \text{if } W_{i,j} > \Delta_j; \\ \alpha_j b_j, & \text{if } |W_{i,j}| \leq \Delta_j; \\ -1, & \text{if } W_{i,j} < -\Delta_j, \end{cases} \tag{20}$$

This also voids hardware efficiency, as the resulting operations are no longer multiplication-free. In contrast, Sherry maintains strict ternary efficiency during inference by utilizing an annealing residual that vanishes post-training.

F Effective Rank for Gradient Analysis

To quantitatively assess the diversity and information density of the gradient updates, we utilize the **Effective Rank** (ER) metric (Roy and Vetterli, 2007). While the standard algebraic rank provides a binary measure of linear independence, it is highly sensitive to infinitesimal noise and fails to capture the numerical stability of the gradient matrix. In contrast, Effective Rank provides a continuous measure of the "learning dimensionality" of a layer by evaluating the entropy of its singular value distribution.

Given a weight gradient matrix $G \in \mathbb{R}^{d_{in} \times d_{out}}$, let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_k)$ denote its singular values obtained via Singular Value Decomposition (SVD), sorted in descending order. We first normalize these values to form a probability-like distribution p:

$$p_i = \frac{\sigma_i}{\sum_{j=1}^{k} \sigma_j}, \quad i = 1, \ldots, k. \tag{21}$$

The Effective Rank is then defined as the exponential of the Shannon entropy of this distribution:

$$ER(G) = \exp\left(-\sum_{i=1}^{k} p_i \ln p_i\right). \quad (22)$$

The value of $ER(G)$ ranges from 1 to k. An ER close to 1 indicates that the gradients are highly *homogenized*, with the update signal collapsing into a single dominant direction. Conversely, an ER close to k suggests a high-rank, diverse update where many independent features are being learned simultaneously.

In the context of N:M sparsity, we use ER to diagnose **Gradient Homogenization**. As shown in Figure 4 and 11, naive 3:4 training regimes often exhibit a relatively small ER as weights become "trapped" in discrete slots, leading to redundant updates. By maintaining a higher ER, Sherry ensures that the sparse ternary model retains the representational richness of its dense counterpart.

G More Experimental Details and Results

G.1 Evaluation Benchmarks

PIQA: The Physical Interaction Question Answering (PIQA) benchmark (Bisk et al., 2020) focuses on physical commonsense reasoning. It tests a model's understanding of everyday physical laws by presenting scenarios (e.g., "How do you stabilize a wobbly table?") and requiring the selection of the correct solution from two options. Success on PIQA indicates a foundational grasp of physical object interactions.

ARC-Easy and ARC-Challenge: The AI2 Reasoning Challenge (ARC) dataset (Clark et al., 2018) is bifurcated to assess scientific knowledge. The ARC-Easy set contains grade-school science questions often answerable via fact retrieval. Conversely, ARC-Challenge consists of questions curated to be difficult for standard retrieval algorithms, requiring complex multi-step reasoning and a deeper conceptual understanding.

HellaSwag: HellaSwag (Zellers et al., 2019) evaluates contextual commonsense inference. Models are given a situational premise and must select the most plausible continuation from four options. Crucially, distractors are adversarially generated to be deceptively plausible, ensuring that success requires a nuanced understanding of event dynamics rather than simple word association.

WinoGrande: WinoGrande (Sakaguchi et al., 2021) is a large-scale dataset for assessing commonsense reasoning through pronoun resolution. Derived from the Winograd Schema Challenge, it presents sentences with ambiguous pronouns and requires the model to determine the correct referent. WinoGrande employs adversarial filtering to reduce statistical biases, forcing models to rely on genuine commonsense understanding.

G.2 Annealing Gate Schedule λ_t

The hyperparameter λ_t plays a crucial role in the annealing gate. Proper scheduling of λ_t from an initial value of 1 to a final value of 0 throughout the training process can significantly improve model performance and convergence stability. This section describes three decay strategies in Fig. 7 for λ_t scheduling, where $p_t \in [0, 1]$ represents the training progress (0: training start, 1: training completion).

The linear decay strategy provides a constant reduction rate of λ_t throughout the training process:

$$\lambda_t = 1 - p_t. \quad (23)$$

The cosine decay strategy follows a convex curve, with slower decay at the beginning and end of training, and faster decay in the middle stage:

$$\lambda_t = \frac{1}{2}\left[1 + \cos(\pi p_t)\right]. \quad (24)$$

The exponential decay strategy follows a concave curve, with rapid decay at the beginning that gradually slows down:

$$\lambda_t = \exp(-5p_t) \quad (25)$$

H Broader Impact

The development of Sherry carries significant implications for the democratization and sustainability of Large Language Models (LLMs). By achieving high-performance 1.25-bit inference through hardware-aligned 3:4 structured sparsity, our work opens up significant societal and technological benefits in resource-constrained scenarios (Li et al., 2025; Huang et al., 2023, 2024a), such as edge/mobile computing (Forman and Zahorjan, 1994; Imielinski and Korth, 1996; Chen et al., 2024a) and cross-device federated learning (McMahan et al., 2017; Huang et al., 2025b). This empowers researchers and developers in resource-constrained environments and developing regions

to access and innovate with large-scale models without prohibitive hardware costs.

There are some potential risks in the application of Sherry. First, extreme quantization (such as our 1.25-bit scheme) is a lossy process that may disproportionately affect the subtle safety guardrails established during fine-tuning (Qi et al., 2023). Quantized models may be more susceptible to jailbreaking or generating harmful content than their full-precision counterparts, as the "safety" representations may be compressed out. Second, cloud-based models benefit from centralized API filtering to prevent malicious use (e.g., generating malware or disinformation). By moving inference entirely offline, Sherry removes this layer of oversight, potentially democratizing access to powerful models for malicious actors without any audit trail. Finally, unlike cloud models, which can be instantly patched to fix safety vulnerabilities, updating models distributed to millions of edge devices is significantly more challenging, potentially leaving vulnerable models in circulation for longer periods.

I Composability with Activation and KV-Cache Quantization

Sherry focuses on weight-only ternary quantization to maximize weight-streaming throughput on edge devices. While 1.25-bit weights drastically reduce the static memory footprint, activations and the KV cache remain in BF16, which constitutes the primary memory bottleneck during long-context or batch inference. Sherry is nonetheless designed to be composable with existing activation and KV-cache quantization techniques:

Activation Quantization. Methods such as SmoothQuant (Xiao et al., 2023) or Quaff (Huang and Wu, 2025) operate by migrating quantization difficulty from activations to weights via channel-wise scaling. These pre-scaling transformations are applied before the matrix multiply and are therefore fully compatible with Sherry's LUT-based inference engine, as long as the activation segments are re-scaled prior to the LUT lookup step.

KV-Cache Compression. Standard INT8/INT4 KV quantization methods are orthogonal to Sherry's weight format and can be stacked directly. A joint setup combining Sherry's 1.25-bit weights with INT4 activations and INT4 KV cache could yield a fully sub-4-bit memory footprint end-to-end, representing a compelling direction for future work.