🐇 RaBiT: Residual-Aware Binarization Training for Accurate and Efficient LLMs

Anonymous authors
Paper under double-blind review

Abstract

Efficient deployment of large language models (LLMs) requires extreme quantization, forcing a critical trade-off between low-bit efficiency and performance. Residual binarization promises hardware-friendly, matmul-free inference by stacking binary (±1) layers, but is plagued by pathological feature **co-adaptation**. We identify a key failure mode, which we term **inter-path adaptation**: during Quantization-Aware Training (QAT), parallel residual binary paths learn redundant features, degrading the error-compensation structure and crippling the model's expressive capacity. While prior work relies on heuristic workarounds (*e.g.,* path freezing) that limit model capacity, we propose **RaBiT**, a novel quantization framework that resolves co-adaptation by algorithmically enforcing a residual hierarchy. Its core mechanism sequentially derives each binary path from a single shared full-precision weight, ensuring each path corrects its predecessor's error. This process is stabilized by a robust initialization that prioritizes functional preservation over mere weight approximation. RaBiT redefines the 2-bit accuracy-efficiency frontier: it achieves state-of-the-art performance, rivals even hardware-intensive Vector Quantization (VQ) methods, and delivers a **4.49× inference speed-up** over full-precision models.

1 Introduction

The massive scale of large language models (LLMs) makes model compression essential for their efficient deployment. While 4-bit quantization methods (Frantar et al., 2023; Lin et al., 2024) have emerged as a successful industry standard (Kwon et al., 2023; Zheng et al., 2024), the relentless pursuit of greater efficiency is pushing the research frontier toward the extreme 2-bit regime. This push toward lower bit compression, however, introduces a critical architectural trade-off that defines the current landscape.

At this frontier, two dominant strategies present a stark choice between accuracy and hardware efficiency. On one hand, Vector Quantization (VQ) methods achieve high accuracy but often introduce hardware overhead from lookup tables or complex rotations (Tseng et al., 2024a;b; Egiazarian et al., 2024). On the other hand, residual binarization—stacking multiple binary layers—offers exceptional, matmul-free efficiency. Yet, this highly efficient approach has consistently struggled to maintain performance, hampered by fundamental training challenges that have prevented it from realizing its full potential (Bulat et al., 2024; Wang et al., 2024; Tran & Nguyen, 2025).

The core promise of a residual architecture—that subsequent paths compensate for the errors of preceding ones—is fundamentally undermined by feature **co-adaptation** (Hinton et al., 2012), a pathological training dynamic where parallel components learn redundant features. In residual binarization, we identify a critical manifestation of this phenomenon, which we term **inter-path adaptation**. During standard Quantization-Aware Training (QAT) (Bengio et al., 2013; Hubara et al., 2018), the structurally agnostic global gradient is applied to all paths simultaneously. This forces them to learn redundant features in a race to minimize the global objective, overriding their intended compensatory roles. The result is a breakdown of the residual hierarchy that severely limits the model's expressive power.

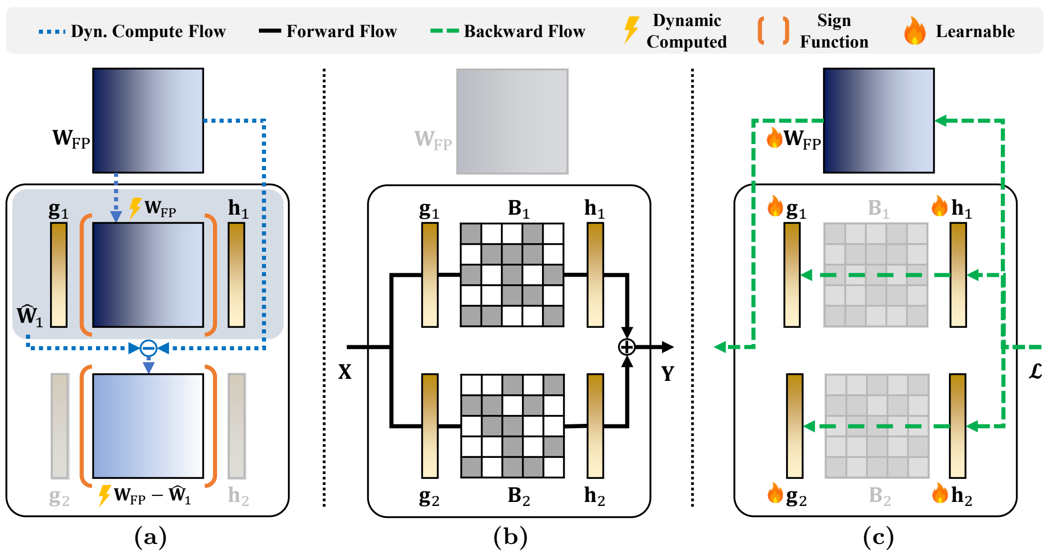


Figure 1: **Overview of the RaBiT training framework. (a) Dynamic Compute Process:** During training, binary paths are dynamically derived from a shared weight $\mathbf{W}_{\text{FP}}$ to enforce a residual hierarchy. **(b) Forward Pass:** For inference, these paths execute in parallel for matmul-free efficiency. **(c) Backward Pass:** Gradients from the loss $\mathcal{L}$ update both the learnable scales $(\mathbf{g}_i, \mathbf{h}_i)$ and the shared $\mathbf{W}_{\text{FP}}$.

Prior attempts to mitigate co-adaptation have relied on heuristic workarounds, such as freezing paths (Bulat et al., 2024; Tran & Nguyen, 2025), which limit the model's capacity to find an optimal joint solution. To address this, we propose **Residual-Aware Binarization Training (RaBiT)**, a QAT framework that resolves inter-path adaptation by design, as depicted in Figure 1. Instead of using independent latent weights, RaBiT maintains a single shared full-precision weight from which binary paths are sequentially derived on-the-fly, guided by learnable scales. This algorithmically enforces the residual hierarchy, training each path to correct its predecessor's error. Combined with a robust, function-aware initialization, RaBiT achieves state-of-the-art accuracy while delivering a 4.49× inference speed-up and halving the training memory footprint.

Our contributions can be summarized as follows:

- We identify and analyze **inter-path adaptation**, a critical manifestation of feature co-adaptation in residual binarization, where the intended error-compensation structure breaks down during Standard QAT as parallel paths become functionally redundant.
- We propose **RaBiT**, a novel QAT framework that resolves inter-path adaptation by enforcing **residual coupling** on-the-fly. The mechanism inherently **halves the training memory footprint** and is stabilized by a robust **function-aware initialization** strategy to tame the unstable dynamics of extreme QAT.
- We demonstrate that RaBiT achieves **state-of-the-art accuracy** at 2-bit precision, delivering a 4.49× inference speed-up while maintaining competitive performance against hardware-intensive VQ methods through matmul-free operations.

2 Related Works

The Shift to QAT in Extreme Quantization. Post-Training Quantization (PTQ) methods, such as GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024), have proven highly successful for compressing large language models to 3- or 4-bit precision by focusing on weight approximation. However, these techniques face a steep performance cliff at lower bit-widths (*e.g.,* 2-bit) (Wang et al., 2023), as the information loss from coarse quantization becomes too severe to overcome by simply minimizing weight reconstruction error. Consequently, the research community is shifting from approximating weights to preserving the model's overall **functionality** (Liu et al., 2025) through Quantization-Aware Training (QAT) (Hubara et al., 2018; Krishnamoorthi, 2018). QAT integrates the simulation of low-precision arithmetic into the fine-tuning process, allowing the model to adapt its parameters to the constraints of the target bit-width. While QAT is challenging due to the non-differentiable nature of quantization—typically addressed with the Straight-Through Estimator (STE) (Bengio et al., 2013)—modern frameworks for binary models have found stability by maintaining a latent full-precision weight for training and updating it via a surrogate gradient (Wang et al., 2023; Xu et al., 2024; Jo et al., 2024; Lee et al., 2025). Our work builds on this robust method to address the unique challenges of residual binary architectures.

Co-adaptation in Residual Binary Architectures. To enhance the limited expressive capacity of a single low-bit layer, residual binarization stacks multiple low-bit paths ($\mathbf{W} \approx \sum_i \hat{\mathbf{W}}_i$) to achieve higher precision while retaining matmul-free efficiency (Wang et al., 2024). However, this parallel architecture is highly susceptible to **feature co-adaptation** (Hinton et al., 2012), a training pathology where components learn redundant features. This phenomenon, which spurred the development of regularization techniques like Dropout (Srivastava et al., 2014).

In the context of residual binarization, we identify a critical form of feature co-adaptation, termed **inter-path adaptation**, where a shared QAT gradient forces parallel paths to learn redundant features, undermining their error-compensation hierarchy. While prior work relied on suboptimal heuristics like path freezing (Bulat et al., 2024; Tran & Nguyen, 2025) that preclude finding a joint optimal solution, RaBiT resolves this core challenge by design, enabling true joint optimization while algorithmically enforcing the hierarchy.

3 Motivation

The goal of Quantization-Aware Training (QAT) is to make a quantized student model, $\mathbf{Y}_s$, functionally mimic its full-precision teacher, $\mathbf{Y}_t$. This is typically achieved by optimizing an objective that combines the final task loss with an intermediate knowledge distillation loss, often formulated as the mean squared error (MSE) (Hinton et al., 2015; Liu et al., 2024).While our full training objective also includes the final KL divergence-based task loss, we focus our analysis on the MSE component for its analytical tractability. The additive structure of the MSE provides a clear window into how parallel paths interact. In a 2-bit residual architecture, the MSE between the teacher output y_t and the student output $y_s = y_1 + y_2$ can be decomposed. Using the Pearson correlation coefficient,[1], this decomposition is:

$$\mathrm{MSE}(y_t, y_s) = \underbrace{(\mathbb{E}[y_t^2] + \mathbb{E}[y_1^2] + \mathbb{E}[y_2^2] - 2\mathbb{E}[y_t y_s])}_{C'} + \underbrace{2\sigma_1\sigma_2}_{PathAmp.} \cdot \underbrace{\mathrm{Corr}(y_1, y_2)}_{PathCorr.}$$

where C' represents the sum of correlation-independent error terms. This reveals a core principle: to minimize the MSE, the paths must be strongly **negatively correlated**. A negative correlation transforms the interaction term into a substantial **bonus** that actively reduces the total loss, signifying effective error-cancellation.

However, Standard QAT structurally fails to achieve this. The shared global gradient induces **inter-path adaptation**, forcing both paths to learn redundant features instead of their intended compensatory roles (see Appendix A.1). To provide a concrete analysis, we decompose the MSE loss for representative layers of Llama2-7B, selected to show the characteristics across the early, middle, and late stages of the network, in Table 1.

The analysis in Table 1 is definitive. Across early, middle, and late stages of the network, the base error term C' and the path amplitude $2\sigma_1\sigma_2$ remain comparable between both methods. The critical difference lies in the correlation. Standard QAT yields a correlation close to zero, resulting in a negligible interaction term that fails to meaningfully reduce the total error. In stark contrast, **RaBiT** structurally enforces a strong negative correlation (*e.g.,*-0.50 in layer 5) (see Appendix A.1). This transforms the interaction term into a significant loss-reducing bonus, systematically lowering the total MSE (see Appendix A.6).

[1]The relationship $\mathbb{E}[y_1 y_2] \approx \sigma_1\sigma_2 Corr(y_1, y_2)$ relies on a zero-mean assumption for the path outputs. We empirically verify this, finding the omitted $\mathbb{E}[y_1]\mathbb{E}[y_2]$ term is less than 1% of the covariance term and thus negligible.

Table 1: **Detailed Decomposition of MSE Loss across Representative Layers of Llama2-7B.** The table breaks down the total MSE into its core components. While the base error (C') and path amplitude ($2\sigma_1\sigma_2$) are comparable, RaBiT consistently generates a strong negative correlation, creating a significant loss-reducing **Bonus**. In contrast, Standard QAT's weak correlation provides a negligible benefit, demonstrating RaBiT's structural advantage in error correction.

Layer	**Method**	**Base Error** (C')	**Path Amp.** ($2\sigma_1\sigma_2$)	**Path Corr.** (Corr)	**Covariance** (Amp. $\times$ Corr)	**Total MSE** (C' + Cov.)
Layer 5 (Early)	Standard QAT	0.0019	0.0030	-0.0752	-0.0002	0.0017
	RaBiT (Ours)	0.0023	0.0028	**-0.4961**	**-0.0014**	**0.0009**
Layer 15 (Mid)	Standard QAT	0.0182	0.0214	-0.1240	-0.0026	0.0156
	RaBiT (Ours)	0.0163	0.0200	**-0.3418**	**-0.0068**	**0.0094**
Layer 25 (Late)	Standard QAT	0.0575	0.0728	-0.1279	-0.0093	0.0482
	RaBiT (Ours)	0.0609	0.0801	**-0.3535**	**-0.0283**	**0.0327**

This principled enforcement of anti-correlation creates a more stable optimization landscape, leading to better generalization and superior performance.

4 Method

We introduce **RaBiT**, a novel QAT framework that prevents interference between the parallel paths of stacked binary architectures. To achieve this, RaBiT enforces a clear error-correction role for each path using a novel **coupled training** loop, and stabilizes the process with a **function-aware initialization** strategy. An overview of the RaBiT training framework is illustrated in Figure 1.

4.1 The Residual Binarization Architecture

To achieve low-bit precision (*e.g.,* 2-bit) while maximizing computational efficiency, we adopt a residual architecture built upon highly efficient binary building blocks.

Binary Building Blocks. The fundamental component is the dual-scale binarization framework. We define the approximation of a weight matrix $\hat{\mathbf{W}}$ using a notation that highlights the underlying element-wise scaling operations:

$$\hat{\mathbf{W}} = \mathbf{g} \odot \mathbf{B} \odot \mathbf{h}. \tag{1}$$

Here, the expression denotes an operation where each element of the resulting matrix, $(\hat{\mathbf{W}})_{ij}$, is computed as $g_i B_{ij} h_j$. $\mathbf{B} \in \{-1, +1\}^{d_{\text{out}} \times d_{\text{in}}}$ is the binary core matrix, and $\mathbf{g} \in \mathbb{R}^{d_{\text{out}}}$, $\mathbf{h} \in \mathbb{R}^{d_{\text{in}}}$ are full-precision, per-channel scaling vectors. The key advantage of this formulation is its matmul-free efficiency. For an input vector $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$, the forward operation computes the output vector $\mathbf{y} \in \mathbb{R}^{d_{\text{out}}}$ as $\mathbf{y} = \mathbf{g} \odot (\mathbf{B}(\mathbf{h} \odot \mathbf{x}))$, which can be implemented using only additions and subtractions, eliminating costly multiplications.

Multi-bit Approximation via Stacking. To enhance representational capacity (*e.g.,* to 2-bit) while retaining this efficiency, we stack $k = 2$ binary paths in parallel. The effective weight is the sum of two binarized terms:

$$\hat{\mathbf{W}}^{(k)} = \sum_{i=1}^{k} \hat{\mathbf{W}}_i = \sum_{i=1}^{k} \mathbf{g}_i \odot \mathbf{B}_i \odot \mathbf{h}_i. \tag{2}$$

This architecture preserves the underlying matmul-free execution, as the forward pass simply accumulates the outputs from each path.

4.2 Coupled Training for Co-Adaptation Mitigation

To address inter-path adaptation, RaBiT abandons the standard approach of training independent latent weights for each binary path. Instead, it maintains a **single shared full-precision (FP) weight $\mathbf{W}_{\text{FP}}$** that serves as the anchor for the entire residual structure.

The Coupled Forward Pass. The core of RaBiT lies in its dynamic forward pass. For a 2-bit architecture with $k = 2$ paths, the binary core matrices, $\mathbf{B}_1$ and $\mathbf{B}_2$, are not stored but re-calculated during every forward pass from the shared weight $\mathbf{W}_{\text{FP}}$ (Figure 1a). This process algorithmically enforces the error-compensation hierarchy. Unlike the dynamically derived binary cores, the scaling vectors $\{\mathbf{g}_i, \mathbf{h}_i\}$ are independent, learnable parameters that capture the magnitude of each path. The derivation proceeds as follows:

1. **Path 1 Derivation**: The first binary core, $\mathbf{B}_1$, is determined by directly binarizing the shared weight: $\mathbf{B}_1 = \text{sign}(\mathbf{W}_{\text{FP}})$. This binary core is then combined with its corresponding learnable scaling vectors, $\mathbf{g}_1$ and $\mathbf{h}_1$, to reconstruct the first-path approximation, $\hat{\mathbf{W}}_1 = \mathbf{g}_1 \odot \mathbf{B}_1 \odot \mathbf{h}_1$.
2. **Residual Calculation**: The residual error, $\mathbf{R}_1$, is calculated by subtracting the *reconstructed* first path from the shared weight: $\mathbf{R}_1 = \mathbf{W}_{\text{FP}} - \hat{\mathbf{W}}_1$.
3. **Path 2 Derivation**: The second binary core, $\mathbf{B}_2$, is then determined by binarizing this freshly computed residual error: $\mathbf{B}_2 = \text{sign}(\mathbf{R}_1)$. The final effective weight used in the forward pass is the sum of the two reconstructed paths: $\hat{\mathbf{W}}^{(2)} = \hat{\mathbf{W}}_1 + (\mathbf{g}_2 \odot \mathbf{B}_2 \odot \mathbf{h}_2)$.

A key design choice is to derive only the **binary cores** $\mathbf{B}_i := \text{sign}(\mathbf{R}_{i-1})$ dynamically, while treating the **scaling vectors** $\{\mathbf{g}_i, \mathbf{h}_i\}$ as independent, learnable parameters. This separation of roles is crucial for both computational efficiency and training stability. Re-calculating optimal scales for the residual at every forward pass—*e.g.,* via Singular Value Decomposition (SVD)—would be prohibitively expensive. Consequently, by maintaining them as learnable parameters, the optimizer can leverage state accumulation (*e.g.,* momentum) to robustly fine-tune the well-initialized values (Section 4.3). This data-adaptive tuning is vital for training stability and allows the error-compensation hierarchy to function effectively by learning the optimal magnitude for each path.

Backward Pass and Parameter Updates. The backward pass is designed for stability and effectiveness. The gradient from the loss $\mathcal{L}$ flows back to update both the independent, learnable scaling vectors $\{\mathbf{g}_i, \mathbf{h}_i\}$ and the single shared weight $\mathbf{W}_{\text{FP}}$, as shown in Figure 1c.

- **Gradient for Learnable Scales**: The scaling vectors $\{\mathbf{g}_i, \mathbf{h}_i\}$ are treated as standard learnable parameters and receive their own gradients via the chain rule. For a mini-batch of size B, the gradients are accumulated over each sample, treating the dynamic binary cores ($\mathbf{B}_i$) as constants during this calculation:

$$\nabla_{\mathbf{g}_i} = \sum_{b=1}^{B} \Delta_b \odot \big(\mathbf{B}_i\,(\mathbf{h}_i \odot \mathbf{X}_b)\big), \quad \nabla_{\mathbf{h}_i} = \sum_{b=1}^{B} \big(\mathbf{B}_i^{\top}(\Delta_b \odot \mathbf{g}_i)\big) \odot \mathbf{X}_b. \tag{3}$$

  Here, b is the sample index within the mini-batch, $\mathbf{X}_b$ is the input vector for that sample, and $\Delta_b = (\partial\mathcal{L}/\partial\mathbf{Y}_b)$ is the upstream gradient from the layer's output $\mathbf{Y}_b$ for that sample.

- **Gradient for the Shared Weight**: To update the single shared weight $\mathbf{W}_{\text{FP}}$, RaBiT uses an **effective-weight gradient**. This acts as a Straight-Through Estimator (STE) for the *entire coupled derivation process*. The gradient is computed with respect to the final effective weight $\hat{\mathbf{W}}^{(k)} = \sum_i \hat{\mathbf{W}}_i$ and is passed back directly to update $\mathbf{W}_{\text{FP}}$:

$$\nabla_{\mathbf{W}_{\text{FP}}} \approx \nabla_{\hat{\mathbf{W}}^{(k)}}\mathcal{L} = \big(\partial\mathcal{L}/\partial\mathbf{Y}\big)^{\top}\mathbf{X}. \tag{4}$$

  In this context, $\mathcal{L}$ is the task loss, while $\mathbf{X}$ and $\mathbf{Y}$ represent the full input and output matrices for the mini-batch. This update completes the training loop: by recomputing the binary paths from the updated $\mathbf{W}_{\text{FP}}$ at every step, RaBiT continuously forces each path to correct the latest residual error, which preserves the overall error-compensation hierarchy.

For inference, the final binary cores $\{\mathbf{B}_i\}$ are derived from the trained $\mathbf{W}_{\text{FP}}$ and then frozen, while the shared weight $\mathbf{W}_{\text{FP}}$ is discarded. The resulting architecture is highly efficient, as the independent binary paths execute in a fully parallel, matmul-free manner. Crucially, this single-weight design also provides a key training advantage: by halving the latent parameters, it reduces the memory required for optimizer states by 50%, a major bottleneck in LLM fine-tuning. The complete training step is detailed in Algorithm 2.

4.3 Stable Initialization for Functional Preservation

QAT in the 2-bit regime is extremely sensitive to the initial quantization error. To mitigate this, we propose a two-stage initialization process that prioritizes preserving model *functionality* over merely approximating weight values.

1. Iterative Residual SVID. The core of our initialization is to find a set of binary paths that jointly approximate a target weight matrix. A standard greedy decomposition is suboptimal because the choice for the first path irreversibly biases all subsequent paths. To find a better joint solution, we propose **Iterative Residual Sign-Value-Independent Decomposition (SVID)**, a Gauss-Seidel style iteration that allows the paths to co-adapt. The process iteratively refines the scales $\{\mathbf{g}_i, \mathbf{h}_i\}$ and binary cores $\{\mathbf{B}_i\}$ for each path $i = 1, \ldots, k$ over $t = 1, \ldots, T$ iterations as follows:

$$\begin{cases} \mathbf{R}_i^{(t)} & := \mathbf{W}_{\text{FP}} - \left(\sum_{j<i} \hat{\mathbf{W}}_j^{(t)} + \sum_{j>i} \hat{\mathbf{W}}_j^{(t-1)}\right), \\ \mathbf{B}_i^{(t)}, \mathbf{g}_i^{(t)}, \mathbf{h}_i^{(t)} & := \text{SVID}(\mathbf{R}_i^{(t)}), \\ \hat{\mathbf{W}}_i^{(t)} & := \mathbf{g}_i^{(t)} \odot \mathbf{B}_i^{(t)} \odot \mathbf{h}_i^{(t)}. \end{cases} \tag{5}$$

Here, SVID$(\cdot)$ (Xu et al., 2024) finds optimal per-channel scales by separating the signs and performing a rank-1 SVD approximation on the magnitudes. Note that this iterative process is not directly applied to the raw weights $\mathbf{W}_{\text{FP}}$, but to a preconditioned target matrix $\mathbf{W}'$, which we define next.

2. I/O Channel Importance-Scaled Preconditioning. To ensure our iterative decomposition focuses on the most functionally critical components of the weight matrix, we do not apply it to the raw weights $\mathbf{W}_{\text{FP}}$. Instead, inspired by recent work on preserving functional saliency (Boža & Hradiš, 2025), we first **precondition** the matrix to create the target $\mathbf{W}'$. Using a small calibration dataset, we compute input activation magnitudes ($\mathbf{s}_{\text{in}}$) and output gradient magnitudes ($\mathbf{s}_{\text{out}}$) and re-weight the full-precision matrix accordingly:

$$\mathbf{W}' \;=\; \mathbf{s}_{\text{out}}^{\alpha_{\text{out}}} \odot \mathbf{W}_{\text{FP}} \odot \mathbf{s}_{\text{in}}^{\alpha_{\text{in}}}. \tag{6}$$

Finally, after running the iterative SVID, the resulting scales are mapped back to the original domain for training: $\mathbf{g}_i = \mathbf{s}_{\text{out}}^{-\alpha_{\text{out}}} \odot \mathbf{g}'_i$ and $\mathbf{h}_i = \mathbf{s}_{\text{in}}^{-\alpha_{\text{in}}} \odot \mathbf{h}'_i$. This strategy dramatically reduces the initial task loss, ensuring a stable start to QAT (see Algorithm 1 for the full algorithm, with analysis in Table 6 and Figure 5).

5 Experiments

5.1 Experimental Settings

Setup. We evaluate RaBiT on recent LLMs including Llama2/3 and Gemma3 (Touvron et al., 2023; AI@Meta, 2024). For QAT, we use a 200M-token subset from a combined WikiText-2 and C4 dataset (Jo et al., 2024). We report perplexity (PPL) on their validation sets (context length: 4096) and the average zero-shot accuracy (QA Avg.) on five common sense reasoning benchmarks (*e.g.,* HellaSwag, PIQA) (Sakaguchi et al., 2021; Zellers et al., 2019; Clark et al., 2018; Bisk et al., 2020), with a detailed breakdown provided in Appendix A.3.

Training Details. We employ a QAT framework with knowledge distillation (KD) (Hinton et al., 2015; Liu et al., 2024), where the full-precision model serves as the teacher. The objective function combines Kullback–Leibler (KL) divergence loss on the output logits with intermediate MSE losses: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{kl}} + \gamma \sum_i \mathcal{L}_{\text{inter},i}$, with $\gamma = 10$ for Llama-family, but $\gamma = 0$ for Gemma3 models to avoid instability from their large activation range. All models are trained for 6 epochs with the Muon optimizer (Jordan et al., 2024) and our proposed function-aware initialization. Full hyperparameters are listed in Appendix A.5.

Baselines. We benchmark RaBiT against a comprehensive set of state-of-the-art 2- to 3-bit methods. Baselines include (1) standard methods like GPTQ and EfficientQAT (Frantar et al.,

Table 2: **Comparison with state-of-the-art 2-3-bit methods on Llama models.** We report perplexity (PPL ↓) and zero-shot QA Average (↑). For the 2-bit results, the best and runner-up are marked in **bold** and <u>underlined</u>, respectively. RaBiT achieves state-of-the-art (SOTA) performance on Llama2-7B and Llama3-8B, while showing highly competitive results on Llama2-13B.

Methods	Llama-2-7B				Llama-2-13B				Llama-3-8B			
	bit	**Wiki2↓**	**C4↓**	**QA Avg↑**	**bit**	**Wiki2↓**	**C4↓**	**QA Avg↑**	**bit**	**Wiki2↓**	**C4↓**	**QA Avg↑**
Baseline	16	5.12	6.63	62.26	16	4.57	6.05	65.46	16	5.75	8.32	68.66
GPTQ	2.1	50.75	36.76	39.16	2.1	43.84	23.07	43.72	2	1.21e3	4.97e2	35.59
EfficientQAT	2.1	6.42	8.34	57.75	2.1	5.58	7.40	62.07	2.1	8.75	12.09	60.63
AQLM	2.3	6.29	8.56	58.57	2.2	5.41	7.20	61.58	2.3	7.23	10.32	64.12
QuIP#	2	6.19	8.16	58.23	2	5.35	7.20	61.96	2	8.70	12.04	63.89
QTIP	2	<u>5.86</u>	<u>7.73</u>	<u>58.97</u>	2	**5.11**	**6.85**	**62.92**	2	<u>7.52</u>	<u>10.76</u>	<u>63.88</u>
BitStack	3	6.91	9.10	56.54	3	5.90	7.86	61.06	3	12.38	17.51	58.41
	2	29.97	34.91	40.12	2	67.98	72.60	39.38	2	2.75e3	1.93e3	36.21
DB-LLM	2	7.23	9.62	55.12	2	6.19	8.38	59.41	2	12.08	16.80	50.92
MBOK	3	6.13	8.13	54.63	3	5.14	6.94	62.73	3	7.81	11.29	61.08
	2	6.99	9.38	53.63	2	5.76	7.89	60.58	2	10.74	14.61	54.41
DBF	2.3	5.81	7.69	59.84	2.3	5.15	6.85	62.53	2.3	7.22	10.34	64.84
	2	6.10	8.05	58.42	2	5.33	7.13	61.53	2	7.78	10.99	62.90
RaBiT(Ours)	3	5.36	7.06	63.05	3	4.84	6.51	64.09	3	6.58	9.54	65.61
	2	**5.77**	**7.64**	**61.51**	2	<u>5.15</u>	<u>6.95</u>	<u>62.10</u>	2	**7.34**	**10.52**	**64.13**

Table 3: **Comparison with state-of-the-art 2-bit methods on Gemma3 models.** We report perplexity (PPL ↓) and zero-shot QA Average (↑). The context length is 4096. RaBiT consistently achieves SOTA or highly competitive performance, demonstrating its robustness across diverse model architectures.

Methods	Gemma3-1B				Gemma3-4B				Gemma3-12B			
	bit	**Wiki2↓**	**C4↓**	**QA Avg↑**	**bit**	**Wiki2↓**	**C4↓**	**QA Avg↑**	**bit**	**Wiki2↓**	**C4↓**	**QA Avg↑**
Baseline	16	9.80	13.69	57.82	16	6.88	10.44	67.60	16	5.50	9.28	73.45
DBF	2	13.28	17.57	51.98	2	8.72	12.71	60.91	2	6.97	10.60	68.37
QTIP	2	13.14	17.36	50.30	2	8.31	12.21	**63.47**	2	**6.65**	10.25	**69.69**
RaBiT(Ours)	2	**11.27**	**15.54**	**53.18**	2	**8.09**	**11.91**	62.21	2	6.66	**10.18**	68.85

2023; Huang et al., 2024); (2) high-accuracy but hardware-intensive Vector Quantization (VQ) approaches such as AQLM, QuIP#, and QTIP (Egiazarian et al., 2024; Tseng et al., 2024a;b); and (3) hardware-efficient binary/residual methods like BitStack, DB-LLM, MBOK, and DBF (Wang et al., 2024; Chen et al., 2024; Tran & Nguyen, 2025; Boža & Hradiš, 2025)[2], which are the most direct architectural competitors.

5.2 Main Results

As shown in Tables 2 and 3, RaBiT consistently redefines the state-of-the-art for 2-bit quantization, demonstrating superior performance across all tested models.

Dominance over Hardware-Efficient Methods. RaBiT significantly outperforms other matmul-free binary/residual methods. On Llama2-7B, its 5.77 WikiText-2 PPL is a marked improvement over competitors like MBOK (6.99 PPL) and DBF (6.10 PPL). This performance gap widens on larger models and more complex datasets, underscoring the severe performance penalty incurred by the **inter-path adaptation** that these methods fail to address. The catastrophic failure of some methods, such as BitStack on Llama3-8B (2.75e3 PPL), highlights the instability that RaBiT's principled design successfully overcomes.

Achieving VQ-Level Accuracy with Binary Efficiency. More impressively, RaBiT closes the gap with and often surpasses hardware-intensive VQ methods, resolving the historical trade-off between accuracy and efficiency. On Llama2-7B, RaBiT's 5.77 PPL edges out the leading VQ method, QTIP (5.86 PPL), setting a new SOTA for 2-bit quantization.

[2]For DB-LLM and MBOK, we used our re-implementation

This trend holds for reasoning tasks, where RaBiT's 61.51% QA average on Llama2-7B surpasses all other listed methods, including QTIP's 58.97%, demonstrating superior functional preservation. RaBiT's robustness is further evident on Llama3-8B, where it maintains strong performance (7.34 PPL) while VQ methods like QuIP# suffer from severe degradation (8.70 PPL), showcasing the stability of our training framework.

5.3 Ablation Studies

5.3.1 Component-wise Contribution Analysis

Channel Importance Scaling (**S**) also yields a consistent, albeit smaller, improvement by prioritizing salient weights. The full RaBiT model, integrating all components, achieves the optimal 5.77 PPL, demonstrating that a residually-coupled training mechanism combined with function-aware initialization is essential for state-of-the-art performance. We performed an ablation study to analyze the contributions of RaBiT's core components: Coupled QAT, Iterative SVID (**I**), and I/O Channel Importance-Scaled Preconditioning (**S**), with results in Table 4.

Table 4: **Ablation on RaBiT** (Llama2-7B PPL). The analysis isolates the impact of Iterative Residual SVID (**I**) and I/O Channel Importance-Scaled Preconditioning (**S**).

Training Method	I	S	Wiki2 ↓
Standard QAT			6.55
	✓		6.21
		✓	6.31
	✓	✓	6.18
Coupled QAT (RaBiT)			5.83
	✓		5.78
		✓	5.81
	✓	✓	**5.77**

The analysis clearly shows that **Coupled QAT** is the most critical performance factor. Simply switching from Standard QAT (6.55 PPL) to Coupled QAT drops the perplexity to 5.83, confirming that resolving inter-path adaptation yields the largest gain.

Our initialization methods (**I** and **S**) provide further essential improvements. While they offer a significant boost to the baseline Standard QAT, their role within the powerful Coupled QAT framework is to provide the final, crucial fine-tuning needed to reach the optimal 5.77 PPL. This synergy between a robust training method and a function-aware initialization is key to RaBiT's state-of-the-art performance.

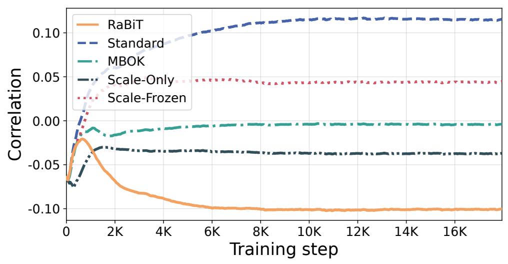
5.3.2 Analysis of Coupled Training Dynamics

To empirically validate that coupled training resolves co-adaptation, we conducted a controlled experiment comparing RaBiT to four variants: (1) **Standard QAT** (independent latent weights), (2) **MBOK** (frozen primary binary core), (3) **Scale-only** (frozen binary cores), and (4) **Scale-frozen** (RaBiT with frozen scales). All variants shared the same initialization and hyperparameters; for a fair comparison of training dynamics, MBOK also used the same optimizer as our model.

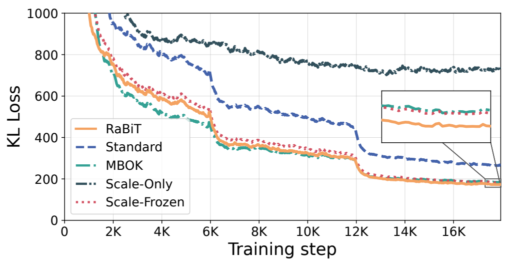
Figure 2 reveals the resulting training dynamics. As theorized, **RaBiT** successfully maintains a stable negative inter-path correlation, enforcing the error-correction hierarchy (Figure 2a). In contrast, **Standard QAT** develops a strong positive correlation, confirming that a shared global gradient induces harmful redundancy. The constrained variants (**MBOK**, **Scale-frozen**) fail to establish a strong anti-correlation, limiting their optimization potential. This structural advantage directly translates to model functionality, as shown by the training loss curves (Figure 2b). RaBiT achieves the lowest and most stable loss, while the co-adaptation in Standard QAT and the incomplete optimization of the other variants lead to significantly higher loss. This analysis confirms that RaBiT's ability to jointly optimize all parameters while algorithmically preventing co-adaptation is the key to its superior performance.

5.4 Inference Performance

RaBiT not only achieves SOTA accuracy but also delivers exceptional inference speed by leveraging its parallelizable matmul-free binary architecture. As shown in Table 5, RaBiT achieves up to a **4.49× speed-up** in end-to-end decoding throughput for a 256-token generation over the FP16 baseline on an NVIDIA RTX 4090.



(a) Inter-Path Correlation



(b) Training Loss

Figure 2: **Visualization of Coupled Training Dynamics. (a) Inter-Path Correlation:** RaBiT enforces a negative inter-path correlation, indicating effective error-correction, whereas Standard QAT leads to positive correlation (co-adaptation). **(b) Training Loss:** This structural advantage directly translates to a lower and more stable training loss for RaBiT, demonstrating its superior optimization path.

This performance gain stems from two key advantages. First, the 8× reduction in model size (2-bit vs. 16-bit) dramatically lowers memory bandwidth requirements, which is the primary bottleneck in the autoregressive decoding phase. Second, unlike VQ methods, RaBiT avoids hardware-unfriendly overheads like lookup tables or rotations. Its simple architecture of additions and element-wise scaling translates to higher hardware utilization. This is evident in our kernel-level benchmarks, where RaBiT's specialized kernels exhibit consistently lower latency than both the FP16 baseline and QTIP's VQ kernels. By eliminating computational complexity, RaBiT ensures that theoretical memory savings translate directly into real-world speed, delivering a solution that is both accurate and genuinely efficient. Further details on our kernel design and additional performance benchmarks are provided in Appendix A.4.1 and Appendix A.4.2, respectively.

Table 5: **Inference Performance Analysis on NVIDIA RTX 4090.** Kernel latency for key Llama2-7B/13B layers and Llama2-7B decoding throughput for a 256-token generation. **RaBiT shows superior efficiency** at both the kernel and system levels.

Method	bit	Kernel-Level Latency (μs) ↓				End-to-End Decoding
		4096×4096 (q_proj, 7B)	11008×4096 (gate_proj, 7B)	5120×5120 (q_proj, 13B)	13824×5120 (gate_proj, 13B)	Throughput (tok/s) ↑
FP16	16	17.15 (1.00×)	70.37 (1.00×)	17.85 (1.00×)	122.90 (1.00×)	64.96 (1.00×)
DBF	2.3	12.66 (1.35×)	28.43 (2.48×)	14.72 (1.21×)	31.87 (3.86×)	157.66 (2.43×)
	2	11.47 (1.50×)	20.90 (3.37×)	14.08 (1.27×)	29.58 (4.15×)	175.21 (2.70×)
QTIP	3	24.04 (0.71×)	37.08 (1.90×)	36.22 (0.49×)	49.97 (2.46×)	153.59 (2.36×)
	2	23.40 (0.73×)	42.40 (1.66×)	37.46 (0.48×)	59.22 (2.08×)	171.74 (2.64×)
RaBiT (Ours)	3	**8.15 (2.10×)**	**17.13 (4.11×)**	**9.90 (1.80×)**	**22.36 (5.50×)**	**191.63 (2.95×)**
	2	**7.72 (2.22×)**	**15.71 (4.48×)**	**8.33 (2.14×)**	**17.50 (7.02×)**	**291.88 (4.49×)**

6 Conclusion

This paper resolves the critical trade-off between accuracy and hardware efficiency in 2-bit LLM quantization by introducing **RaBiT**. We first identify and analyze **inter-path adaptation** as a fundamental bottleneck that compromises the error-compensation structure in residual binarization. RaBiT's core mechanism, on-the-fly **residual coupling**, algorithmically prevents this breakdown during training, ensuring the model's full expressive power is utilized. To tame the notoriously unstable dynamics of extreme quantization, we further introduce a robust **function-aware initialization** strategy that ensures stable convergence. Our comprehensive experiments demonstrate that RaBiT achieves new **state-of-the-art performance** at 2-bit precision, surpassing not only existing binary methods but also complex and hardware-intensive vector quantization approaches. By establishing this new frontier, RaBiT paves the way for the efficient deployment of high-performance low-bit LLMs and provides a scalable foundation for future research.

Reproducibility Statement

We are committed to ensuring the reproducibility of our research. To this end, we provide detailed descriptions of our methodology, experimental setup, and hyperparameters throughout the paper and its appendices.

Algorithm and Implementation Details. While full source code availability is subject to our organization's review protocols, we have made every effort to ensure reproducibility by providing detailed algorithmic descriptions. The core logic of the **RaBiT** QAT framework is thoroughly explained in Section 4 and presented as step-by-step pseudocode in Algorithm 2. Similarly, the design and optimization principles of our high-performance CUDA kernel are described in Appendix A.4.1, providing a clear blueprint for implementation.

Data. Our training process utilizes a 200M-token subset of the publicly available WikiText-2 and C4 datasets, generated using the same data processing approach as described in Jo et al. (2024)[3]. All evaluations are performed on standard public benchmarks (WikiText-2, C4, HellaSwag, PIQA, ARC-e, ARC-c, and WinoGrande), as detailed in Section 5.1.

Hyperparameters and Infrastructure. A comprehensive list of all hyperparameters used for training each model, including model-specific learning rates, optimizer settings, and the initialization parameters ($\alpha_{\text{in}}, \alpha_{\text{out}}, T_{\max}$), is provided in Appendix A.5. All experiments were conducted on a single node equipped with four NVIDIA H100 GPUs. The reported results are from a single training run for each model, which is a standard practice for LLM fine-tuning at this scale due to the high computational cost.

References

AI@Meta. The Llama 3 herd of models. arXiv:2407.21783, 2024.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv:1308.3432, 2013.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

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

Adrian Bulat, Yassine Ouali, and Georgios Tzimiropoulos. QBB: Quantization with binary bases for llms. *Advances in Neural Information Processing Systems*, 37:3209–3228, 2024.

Hong Chen, Chengtao Lv, Chaofan Yang, Yuxuan Zhang, Zhong-Qiu Wang, Heng Zhang, Jie Chen, Enhong Chen, and Dacheng Tao. DB-LLM: Accurate dual-binarization for efficient LLMs. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

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

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[3]The specific code we used is available in the official repository: `https://github.com/dongwonjo/BinaryMoS/blob/main/utils/datautils.py`.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv:1503.02531, 2015.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Hanxian Huang, Zechun Liu, Changsheng Zhao, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. EfficientQAT: A block-wise quantization-aware fine-tuning framework for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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

Dongwon Jo, Taesu Kim, Yulhwa Kim, and Jae-Joon Kim. Mixture of Scales: Memory-efficient token-adaptive binarization for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Banseok Lee, Dongkyu Kim, Youngcheon You, and Youngmin Kim. Littlebit: Ultra low-bit quantization via latent factorization. *arXiv preprint arXiv:2506.13771*, 2025.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. *Proceedings of Machine Learning and Systems (MLSys)*, 6:87–100, 2024.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2024.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, et al. ParetoQ: Scaling laws in extremely low-bit llm quantization. *arXiv preprint arXiv:2502.02631*, 2025.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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

Ba-Hien Tran and Van Minh Nguyen. Highly efficient and effective llms with multi-boolean architectures. *arXiv preprint arXiv:2505.22811*, 2025.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024a.

Albert Tseng, Mark Z Mao, Jerry Chee, Tamas Linder, Volodymyr Kuleshov, Robert M Gray, and Christopher De Sa. QTIP: Quantization with trellises and incoherence processing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

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

Xinghao Wang, Pengyu Wang, Bo Wang, Dong Zhang, Yunhua Zhou, and Xipeng Qiu. Bitstack: Any-size compression of large language models in variable memory environments. *arXiv preprint arXiv:2410.23918*, 2024.

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.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

Feng Zhang, Yanbin Liu, Weihua Li, Jie Lv, Xiaodan Wang, and Quan Bai. Towards superior quantization accuracy: A layer-sensitive approach. *arXiv preprint arXiv:2503.06518*, 2025.

Yang Zhang, Yanfei Dong, and Kenji Kawaguchi. Investigating layer importance in large language models. *arXiv preprint arXiv:2409.14381*, 2024.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. SGLang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

A Appendix

A.1 Mathematical Analysis of Training Dynamics

This section provides a mathematical analysis of the training dynamics for residual binary architectures. We demonstrate why Standard QAT is prone to **inter-path adaptation**, where paths become redundant. In contrast, we show how RaBiT's coupled training mechanism structurally enforces an **error-correcting hierarchy** and is superior to other heuristic solutions.

Proposition 1 (Inter-Path Adaptation in Standard QAT)**.** *In a Standard QAT scheme where two paths ($\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2$) are updated from their respective latent weights ($\mathbf{W}_1, \mathbf{W}_2$) using a shared global gradient $\mathbf{G} = \nabla_{\hat{\mathbf{W}}_1+\hat{\mathbf{W}}_2}\mathcal{L}$, the paths have a persistent tendency to become positively correlated, leading to redundancy.*

Proof. Let the latent weights be $\mathbf{W}_1$ and $\mathbf{W}_2$. After a single update step with learning rate η and shared gradient $\mathbf{g}$, the new weights are $\mathbf{W}_1'$ and $\mathbf{W}_2'$:

$$\mathbf{W}_1' := \mathbf{W}_1 - \eta\mathbf{G} \quad \text{and} \quad \mathbf{W}_2' := \mathbf{W}_2 - \eta\mathbf{G}$$

The change in the Frobenius inner product between the weights, which reflects their correlation, is:

$$\Delta_{\langle\cdot,\cdot\rangle} := \langle \mathbf{W}_1', \mathbf{W}_2' \rangle_F - \langle \mathbf{W}_1, \mathbf{W}_2 \rangle_F$$

Expanding this gives:

$$\Delta_{\langle\cdot,\cdot\rangle} = -\eta\left(\langle \mathbf{W}_1, \mathbf{G} \rangle_F + \langle \mathbf{W}_2, \mathbf{G} \rangle_F\right) + \eta^2\|\mathbf{G}\|_F^2$$

While the linear terms depend on the alignment between the current weights and the gradient, the quadratic term $\eta^2\|\mathbf{G}\|_F^2$ is **always non-negative**. This term acts as a systematic force, constantly pushing the two paths in the same direction defined by the global gradient $\mathbf{G}$. This dynamic, the underlying mechanism of **inter-path adaptation**, compels both paths to learn redundant, dominant features in order to minimize the global loss. This ultimately leads to a break down of the intended residual hierarchy and compromises the model's expressive capacity. □

Proposition 2 (Structurally Enforced Error Correction in RaBiT)**.** *RaBiT's coupled training mechanism resolves the redundancy drift by fundamentally changing the optimization objective. Instead of independent updates, RaBiT's on-the-fly derivation structurally forces the second path ($\hat{\mathbf{W}}_2$) to align with the true residual of the first path ($\mathbf{R}_1 = \mathbf{W}_{\text{FP}} - \hat{\mathbf{W}}_1$), thereby enforcing an error-correcting relationship.*

Analysis. The optimization objectives of the two paths are implicitly different in RaBiT versus the naïve approach.

- **Standard QAT Objective**: Both paths are driven by the same structurally-agnostic global gradient $\mathbf{G}$. Their implicit goal is to align with $\mathbf{G}$ to reduce the global loss. Since both $\hat{\mathbf{W}}_1$ and $\hat{\mathbf{W}}_2$ are incentivized to align with the same vector $\mathbf{G}$, they inevitably learn to align with each other, leading to redundancy as shown in Proposition 1.

$$\hat{\mathbf{W}}_1 \propto \mathbf{G} \quad \text{and} \quad \hat{\mathbf{W}}_2 \propto \mathbf{G} \implies \langle \hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2 \rangle_F > 0$$

- **RaBiT's Enforced Objective**: RaBiT maintains a single shared blueprint, $\mathbf{W}_{\text{FP}}$. The on-the-fly derivation process, $\mathbf{R}_1 := \mathbf{W}_{\text{FP}} - \hat{\mathbf{W}}_1$ followed by the binarization of $\mathbf{R}_1$ to create $\hat{\mathbf{W}}_2$, explicitly defines the optimization target for the second path. The goal for $\hat{\mathbf{W}}_2$ is no longer to align with the global gradient $\mathbf{g}$, but to be the best possible low-rank approximation of the current residual $\mathbf{R}_2$.

$$\text{Objective for } \hat{\mathbf{W}}_2: \quad \min \|\mathbf{R}_1 - \hat{\mathbf{W}}_2\|_F^2 \implies \hat{\mathbf{W}}_2 \approx \mathbf{R}_1$$

  This structural constraint forces a high **Residual Alignment**. In the context of extreme low-bit quantization, the first approximation $\hat{\mathbf{W}}_1$ often "overshoots" the target $\mathbf{W}_{\text{FP}}$ in certain directions. To correct this, the residual $\mathbf{R}_1 = \mathbf{W}_{\text{FP}} - \hat{\mathbf{W}}_1$ will point in the opposite direction of the overshoot. By aligning with $\mathbf{R}_1$, $\hat{\mathbf{W}}_2$ naturally becomes **anti-correlated** with $\hat{\mathbf{W}}_1$, implementing an efficient **active cancellation** mechanism rather than degenerating into redundancy.

□

Proposition 3 (Superior Optimization Dynamics of Coupled vs. Iterative Training)**.** *Iterative training (e.g., freezing one path while training the other) avoids adaptation but at the cost of optimization efficiency. In contrast, RaBiT resolves adaptation while permitting full parameter co-adaptation, resulting in a superior optimization trajectory.*

Proof. Following Proposition 1, the problem of Standard QAT is the simultaneous update of both paths in the same direction. An alternative solution is to update them iteratively, which prevents this simultaneous push and thus avoids adaptation. However, this introduces a new problem of inefficiency.

The optimal direction to reduce the loss $\mathcal{L}$ is the steepest descent direction in the joint parameter space of $(\mathbf{W}_1, \mathbf{W}_2)$, which is $\mathbf{d}^* = (-\mathbf{G}, -\mathbf{G})$. When training iteratively, one path is frozen, so the update is restricted to an axis-aligned direction, e.g., $\mathbf{d}_{\text{iter}} = (\mathbf{0}, -\mathbf{G})$. The cosine similarity between the iterative update and the optimal update direction is:

$$\cos(\theta) = \frac{\langle \mathbf{d}_{\text{iter}}, \mathbf{d}^* \rangle_F}{\|\mathbf{d}_{\text{iter}}\|_F \|\mathbf{d}^*\|_F} = \frac{\langle (\mathbf{0}, -\mathbf{G}), (-\mathbf{G}, -\mathbf{G}) \rangle_F}{\|(\mathbf{0}, -\mathbf{G})\|_F \|(-\mathbf{G}, -\mathbf{G})\|_F}$$
$$= \frac{\|\mathbf{G}\|_F^2}{\|\mathbf{G}\|_F \cdot \sqrt{\|\mathbf{G}\|_F^2 + \|\mathbf{G}\|_F^2}} = \frac{1}{\sqrt{2}}$$

This fixed 45° misalignment forces the optimization to follow an inefficient zig-zag trajectory. While it solves adaptation, it sacrifices optimization efficiency.

RaBiT, through its coupled derivation described in Proposition 2, resolves this trade-off. By updating a single shared weight $\mathbf{W}_{\text{FP}}$ with the full gradient $\mathbf{G}$, it allows both paths to co-adapt simultaneously in a coordinated manner that is not restricted to an inefficient path. Thus, RaBiT resolves adaptation without compromising optimization efficiency, leading to superior dynamics. □

Corollary 1 (to Proposition 2). Negative Correlation Induction. *RaBiT's coupled training mechanism, by forcing the second path ($\hat{\mathbf{W}}_2$) to approximate the residual of the first path ($\mathbf{R}_1$), inherently promotes a negative correlation between their respective outputs ($\mathbf{y}_1, \mathbf{y}_2$).*

Analysis. From Proposition 2, we established that RaBiT trains the second path to approximate the residual of the first:

$$\hat{\mathbf{W}}_2 \approx \mathbf{R}_1 = \mathbf{W}_{\text{FP}} - \hat{\mathbf{W}}_1$$

Let's consider the outputs for a given input $\mathbf{x}$. The outputs of the full-precision teacher, the first path, and the second path are $\mathbf{y}_t = \mathbf{W}_{\text{FP}}\mathbf{x}$, $\mathbf{y}_1 = \hat{\mathbf{W}}_1\mathbf{x}$, and $\mathbf{y}_2 = \hat{\mathbf{W}}_2\mathbf{x}$, respectively. Based on the weight approximation, the output of the second path is:

$$\mathbf{y}_2 \approx \mathbf{R}_1\mathbf{x} = (\mathbf{W}_{\text{FP}} - \hat{\mathbf{W}}_1)\mathbf{x} = \mathbf{y}_t - \mathbf{y}_1$$

Now, we can analyze the covariance between the outputs $\mathbf{y}_1$ and $\mathbf{y}_2$. Assuming the outputs are centered for simplicity, the covariance is proportional to the expected value of their dot product, $\mathbb{E}[\mathbf{y}_1^\top \mathbf{y}_2]$.

$$\mathbb{E}[\mathbf{y}_1^\top \mathbf{y}_2] \approx \mathbb{E}[\mathbf{y}_1^\top (\mathbf{y}_t - \mathbf{y}_1)] = \mathbb{E}[\mathbf{y}_1^\top \mathbf{y}_t] - \mathbb{E}[\mathbf{y}_1^\top \mathbf{y}_1] = \mathbb{E}[\mathbf{y}_1^\top \mathbf{y}_t] - \mathbb{E}[\|\mathbf{y}_1\|^2]$$

Let's analyze the two terms:

1. $\mathbb{E}[\mathbf{y}_1^\top \mathbf{y}_t]$: The first path $\hat{\mathbf{W}}_1$ is the primary, albeit coarse, approximation of $\mathbf{W}_{\text{FP}}$. Its purpose is to capture the main features of the teacher. Therefore, their outputs $\mathbf{y}_1$ and $\mathbf{y}_t$ are expected to be strongly and **positively correlated**, making this term a large positive value.

2. $\mathbb{E}[\|\mathbf{y}_1\|^2]$: This is the expected squared norm of the first path's output. Binarization is an aggressive quantization that often leads to an "overshoot" in magnitude. A single binary path must represent a wide range of continuous values, so its effective scaling factor often results in an output magnitude $\|\mathbf{y}_1\|$ that exceeds the projection of $\mathbf{y}_1$ onto $\mathbf{y}_t$. Consequently, it is generally the case that $\|\mathbf{y}_1\|^2 > \mathbf{y}_1^\top \mathbf{y}_t$, making $\mathbb{E}[\|\mathbf{y}_1\|^2]$ a larger positive term than $\mathbb{E}[\mathbf{y}_1^\top \mathbf{y}_t]$.

Combining these points, the covariance is approximately the difference between a positive term and a larger positive term:

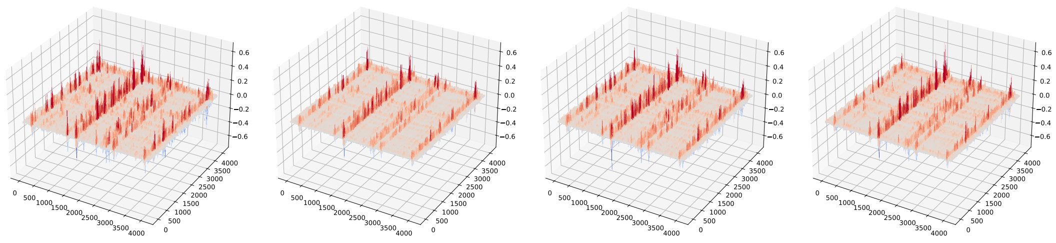
$$\mathrm{Cov}(\mathbf{y}_1, \mathbf{y}_2) \approx \underbrace{\mathbb{E}[\mathbf{y}_1^\top \mathbf{y}_t]}_{\text{Positive Alignment}} - \underbrace{\mathbb{E}[\|\mathbf{y}_1\|^2]}_{\text{Larger Magnitude Term}} < 0$$

Thus, RaBiT's mechanism of forcing the second path to correct the error of the first path structurally drives the covariance, and therefore the correlation $\mathrm{Corr}(\mathbf{y}_1, \mathbf{y}_2)$, to be negative. $\square$

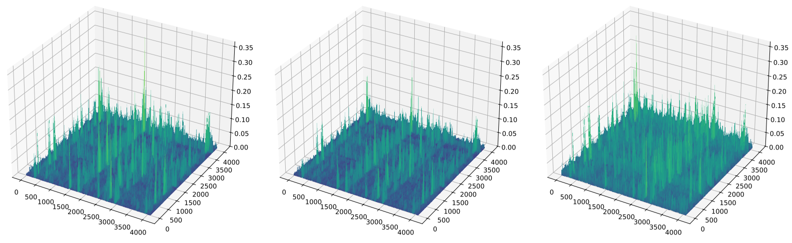
A.2 Initialization Analysis: Functionality vs. Approximation

Table 6: **Initialization Analysis on Llama2-7B.** Trade-off between weight reconstruction error (Avg. MAE/MSE) and model functionality (Initial KL Divergence Loss), for the first `q_proj` layer. I/O Channel Importance Scaling dramatically reduces KL Divergence Loss despite increasing MSE.

Initialization Method	**Avg. MAE ↓**	**Avg. MSE ↓**	**KL Loss ↓**
Greedy SVID	0.359	0.150	17,152
Iterative Residual SVID	0.370	0.122	13,760
+ I/O Ch. Importance Scaling	**0.632**	**0.302**	**2,672**



(a) $\mathbf{W}_{\mathrm{FP}}$ (b) Greedy SVID Init. (c) Iterative SVID Init. (d) Iter + I/O Scaling



(e) Diff. (Greedy) (f) Diff. (Iterative) (g) Diff. (Iter + I/O)

Figure 3: **Visual analysis of weight initialization for the first layer's `q_proj` matrix of the Llama2-7B model.** The top row displays the original full-precision weight ($\mathbf{W}_{\mathrm{FP}}$) alongside its initial approximations from three methods: (b) Greedy SVID, (c) Iterative SVID, and (d) Iterative SVID with I/O Channel Importance Scaling. The bottom row shows the corresponding difference matrices ($\mathbf{W}_{\mathrm{FP}} - \hat{\mathbf{W}}_{\mathrm{init}}$), illustrating the initial error structure. Our function-aware initialization produces a qualitatively different structure compared to the others, which is reflected in its distinct error pattern.

Stable initialization is paramount in the low-bit regime, as the initial quantization error spike can destabilize QAT. On the Llama2-7b model, we evaluate our proposed techniques—Iterative Residual SVID and I/O Channel Importance Scaling (Section 4.3)—by measuring both the weight reconstruction error (Avg. MAE, MSE) and the initial task loss (Knowledge Distillation (KD) loss) before the first training step.

Table 6 details the results averaged across attention projection layers and reveals a crucial insight. As our baseline, Greedy SVID is a non-iterative decomposition that finalizes each path sequentially without the co-adaptation enabled by our iterative approach. First, regarding **Iterative Refinement**, moving from Greedy SVID to Iterative Residual SVID consistently improves weight reconstruction (*e.g.,* Avg. MSE drops 0.150 → 0.122) and substantially reduces the initial KL divergence loss (17,152 → 13,760), confirming mitigation of scheduling bias. Second, adding **I/O Channel Importance Scaling** to the iterative process yields a striking result: while reconstruction error increases significantly (Avg. MSE 0.122 → 0.302), the KL divergence loss *plummets* dramatically (13,760 → 2,672, an 81% reduction).

This confirms that extreme quantization should prioritize preserving *functionality* over merely approximating *weights*. I/O Channel Importance Scaling allocates the limited 2-bit capacity to critical channels based on activation and gradient statistics (Section 4.3), sacrificing the reconstruction of less important weights. This trade-off is visually stark in Figure 3. While Iterative SVID produces a lower-error approximation than Greedy SVID (comparing Figure 3f to Figure 3e), the function-aware I/O Scaling method yields a visibly larger reconstruction error (Figure 3g). Despite this higher weight-level discrepancy, its focus on functional saliency provides a far superior starting point for QAT, as evidenced by the dramatic reduction in initial task loss.

A.3 Extended Results

Detailed Zero-Shot Reasoning Accuracy. Table 7 and Table 8 provide a detailed breakdown of the zero-shot reasoning accuracy across five common benchmarks, complementing the average scores reported in the main text.

Table 7: **Detailed Zero-Shot Reasoning Accuracy on Llama Models** (%). Comparison of FP16 against leading 2-bit methods on five common benchmarks.

Models	Method	WinoGrande↑	HellaSwag↑	ARC-e↑	ARC-c↑	PIQA↑	Average↑
Llama2-7B	FullPrecision	67.80	56.71	69.28	39.93	78.29	62.40
	QTIP	64.64	53.09	65.57	35.67	**75.90**	58.97
	DBF	63.61	52.44	64.73	35.58	75.84	58.44
	RaBiT (Ours)	**67.80**	**53.52**	**72.43**	**37.88**	**75.90**	**61.51**
Llama2-13B	FullPrecision	69.93	59.64	73.19	45.73	78.67	65.43
	QTIP	**67.56**	**57.4**	**70.8**	**41.46**	77.37	**62.92**
	DBF	67.09	56.6	69.02	38.74	78.18	61.93
	RaBiT (Ours)	**67.56**	56.71	69.06	39.76	**77.42**	62.10
Llama3-8B	FullPrecision	72.93	60.08	80.30	50.17	79.76	67.80
	QTIP	**70.24**	**55.53**	75.29	41.64	76.71	63.88
	DBF	68.90	54.49	74.62	39.76	76.44	62.84
	RaBiT (Ours)	69.37	55.13	**75.37**	**42.83**	**77.96**	**64.13**

Table 8: **Detailed Zero-Shot Reasoning Accuracy on Gemma Models** (%). Comparison of FP16 against leading 2-bit methods on five common benchmarks.

Models	Method	WinoGrande↑	HellaSwag↑	ARC-e↑	ARC-c↑	PIQA↑	Average↑
Gemma3-1B	FullPrecision	59.59	47.30	72.22	35.32	74.65	57.82
	QTIP	54.62	38.24	63.93	25.85	68.88	50.30
	DBF	**58.01**	40.37	62.92	28.41	70.18	51.98
	RaBiT (Ours)	56.59	**42.94**	**64.52**	**29.44**	**72.42**	**53.18**
Gemma3-4B	FullPrecision	69.22	56.77	81.52	51.45	79.05	67.60
	QTIP	**66.85**	52.25	**77.53**	**44.62**	76.12	**63.47**
	DBF	63.69	50.15	74.74	40.87	75.08	60.91
	RaBiT (Ours)	65.19	**52.57**	75.04	41.38	**76.88**	62.21
Gemma3-12B	FullPrecision	75.45	61.98	87.08	61.60	81.12	73.45
	QTIP	**72.69**	57.99	**84.09**	**54.95**	78.73	**69.69**
	DBF	72.14	57.20	82.49	52.05	77.97	68.37
	RaBiT (Ours)	72.30	**58.45**	82.41	52.13	**78.95**	68.85

A.4 Inference Performance Analysis

A.4.1 Kernel Design

Our CUDA kernels implement binary GEMV operations tailored to the memory-bound regime typical of the decoding phase in LLM inference. The design centers on bit-packing to reduce global memory traffic, with a latency-tolerant and matmul-free compute pipeline that leverages register-level staging.

Weight Packing. To reduce memory traffic, each group of 32 columns is mapped to a `uint32_t`, with $+1 \mapsto 0$ and $-1 \mapsto 1$. We then group the 32-bit words into `uint2` or `PackedBits3` (3 `uint32_t` weights with padding), for 2-bit (2 binary weights) and 3-bit (3 binary weights) models, respectively. Rows are interleaved into warp-sized groups, ensuring that a warp issues full coalesced memory transactions when loading weights. Our efficient packing reduces the raw footprint of weights by a factor of $32\times$, compared to full-precision weights.

Compute Pipeline. Each warp is assigned a set of output rows to avoid inter-warp synchronization. Input activations ($\mathbf{x}$) and column scales ($\mathbf{g}$) are read as vectorized `uint4` chunks. Binary signs are applied via lane-local bit shifts and XOR masks, instead of matrix multiplication. The kernel uses simple yet effective pipelining: while one tile of data is consumed, the subsequent tile is prefetched into registers. Accumulation proceeds using `half2` fused multiply-add intrinsics (`__hfma2`), which increase arithmetic throughput without resorting to shared memory. Finally, reductions across threads in a warp are performed with shuffle operations, and output scale factors ($\mathbf{h}$) are applied in `fp16` precision. Notably, our architecture enables per-path parallelizable computation - instead of an n-bit weight, we parallelize with n 1-bit operations.

Efficient weight packing and pipelining reduce global memory access and raise the utilization of execution units on the GPU. The kernel therefore shifts the limiting factor from raw memory bandwidth toward register throughput, yielding measurable efficiency gains during the decoding stage of LLM inference, showing remarkable performance.

A.4.2 More Comparisons

To provide a more detailed analysis of the end-to-end inference speed, we benchmarked RaBiT against several key baselines: the full-precision (FP16) model, QTIP as the state-of-the-art Vector Quantization (VQ) method, and DBF, which features a similar stacked binary architecture. For QTIP, we utilized the publicly available CUDA kernels from the official implementation[4]. For DBF, which also uses a stacked binary design but executes its two paths sequentially, we developed an optimized CUDA kernel that runs approximately 21% faster than their public Triton-based implementation to ensure a fair and robust comparison. All evaluations were conducted on an NVIDIA RTX 4090 using the Llama2-7B model.

The results, depicted in Figure 4, were benchmarked across a range of generated token lengths (64, 128, 256, 512, and 1024) for a comprehensive analysis. As expected, all 2-bit methods significantly outperform the FP16 baseline due to the $8\times$ reduction in memory bandwidth requirements. More importantly, **RaBiT demonstrates a substantial performance advantage, achieving nearly twice the decoding throughput** of the other 2-bit quantization methods. This speed-up stems directly from the efficiency of our parallel, matmul-free architecture. Unlike DBF, which is bottlenecked by its sequential computation of two binary paths, RaBiT's fully parallel design allows it to maximally leverage the benefits of its efficient binary cores. While the absolute tokens/second rate naturally decreases with longer generation sequences, the relative performance gap between the methods remains consistent, confirming the robustness of RaBiT's architectural advantage.

[4]`https://github.com/Cornell-RelaxML/qtip`

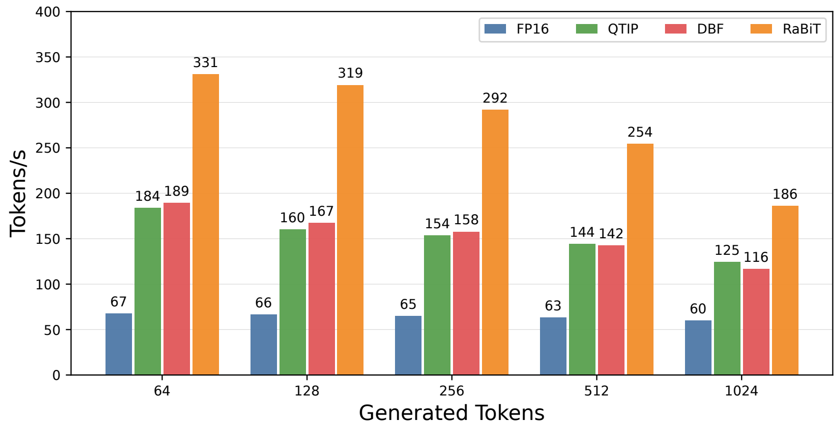


Figure 4: **End-to-end decoding throughput (tokens/second) for Llama2-7B on an NVIDIA RTX 4090 across various generated token lengths.** RaBiT's parallel architecture consistently delivers superior performance over other 2-bit methods.

A.5 Hyperparameters

A.5.1 Training Details

We detail the hyperparameters used for our Quantization-Aware Training (QAT) experiments in Table 9. All models were trained for 6 epochs using the Muon optimizer (Jordan et al., 2024) with a cosine learning rate decay schedule. The models were initialized using our proposed function-aware strategy, with a fixed SVID iteration count of $T_{\max} = 20$. Key hyperparameters, such as the learning rate and the I/O Channel Importance Scaling intensities $(\alpha_{\text{in}}, \alpha_{\text{out}})$, were fine-tuned for each specific model to achieve the best performance. All experiments were conducted on a single node equipped with four NVIDIA H100 GPUs.

Table 9: **RaBiT** Training Details

	Training Setup	Llama2		Llama3	Gemma3		
Bit	Target	7B	13B	8B	1B	4B	12B
2	Intensities $(\alpha_{\text{in}}, \alpha_{\text{out}})$	(0.8, 0.65)	(0.95, 0.45)	(0.85, 0.7)	(0.85, 0.7)	(0.95, 0.7)	(0.75, 0.6)
	Iteration $(T_{\max})$	20	20	20	20	20	20
	Learning Rate	12e-6	1e-5	1e-5	1e-5	1e-5	5e-6
	Epoch	6	6	6	6	6	6
	# GPUs	1×4	1×4	1×4	1×4	1×4	1×4
	# Training Hours	39	56	38	8	23	67
3	Intensities $(\alpha_{\text{in}}, \alpha_{\text{out}})$	(0.8, 0.65)	(0.95, 0.45)	(0.85, 0.7)	-	-	-
	Iteration $(T_{\max})$	20	20	20	-	-	-
	Learning Rate	1e-5	1e-5	1e-5	-	-	-
	Epoch	6	6	6	-	-	-
	# GPUs	1×4	1×4	1×4	-	-	-
	# Training Hours	46	88	44	-	-	-

A.5.2 Grid Search for I/O Channel Importance Scaling Intensities

To determine the optimal intensity hyperparameters for our I/O Channel Importance Scaling (Section 4.3), we performed a comprehensive grid search. The objective was to identify the values of α_{in} and α_{out} that minimized the initial Knowledge Distillation (KD) loss post-initialization. This process utilized a calibration dataset of 128 samples randomly selected from the training data to measure the loss.

The example results of this search on the Llama2-7B model are detailed in Table 10. We observed a clear optimum, with the minimum initial KL divergence loss of 2,672 achieved at the configuration of $\alpha_{\text{in}} = 0.80$ and $\alpha_{\text{out}} = 0.65$. This finding underscores the importance of a balanced preconditioning strategy that considers both input activation statistics and output gradient magnitudes. **We repeated this grid search process for all other models to find their optimal alpha values.**

Table 10: **Grid search results for I/O Channel Importance Scaling Intensities (α_{in}, α_{out}) on Llama2-7B.** The metric is the Initial KL Divergence Loss (Lower is better). The optimal configuration is highlighted in bold.

α_{out}	α_{in} 0.75	**0.80**	0.85	0.90
0.55	3,100	2,932	2,984	3,108
0.60	2,932	2,938	2,971	3,143
0.65	3,167	**2,672**	2,697	3,063
0.70	3,083	2,821	2,983	3,462

A.5.3 SVID Iteration Convergence Analysis

The Iterative Residual SVID initialization (Section 4.3) aims to mitigate the scheduling bias inherent in standard greedy initialization. We analyzed the required number of iterations ($T_{\max}$) for convergence on the Llama2-7B model. We measured the Initial KL divergence loss as the iterations progressed from 1 (equivalent to Greedy SVID) up to 35.

The results, shown in Figure 5, indicate that the initialization quality improves rapidly in the initial phase. The loss stabilizes significantly around 15 iterations, and the optimum is reached at 20 iterations. Beyond this point, further iterations do not provide additional benefits. Based on this analysis, we selected $T_{\max} = 20$ as the default setting for RaBiT initialization, providing an optimal balance between initialization quality and computational cost.

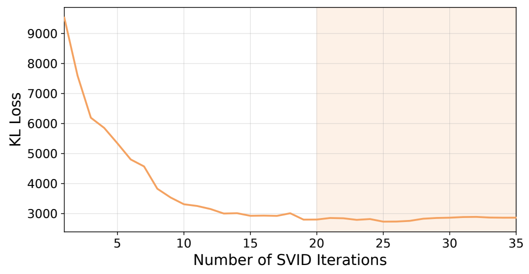


Figure 5: **Convergence analysis of Iterative Residual SVID on Llama2-7B**. The metric is the Initial KL Divergence Loss (Lower is better). Convergence stabilizes around 20 iterations.

A.6 Extended Analysis of Inter-Path Adaptation

To provide a more granular view of the training dynamics, we conduct a layer-wise analysis of the Mean Squared Error (MSE) decomposition for the Llama2-7B model, visualized in Figure 6. This analysis offers two key insights into RaBiT's structural advantages over Standard QAT.

First, the results empirically confirm our central hypothesis across the network's depth. For most layers, RaBiT consistently generates a substantial negative covariance (the red-dashed component), which acts as a significant loss-reducing bonus, thereby lowering the total MSE. In contrast, Standard QAT fails to establish this effective error-cancellation, exhibiting a much smaller covariance term that provides negligible benefit. This provides strong visual evidence that RaBiT's coupled training successfully enforces the intended error-correction hierarchy, while Standard QAT suffers from the performance degradation of inter-path adaptation.

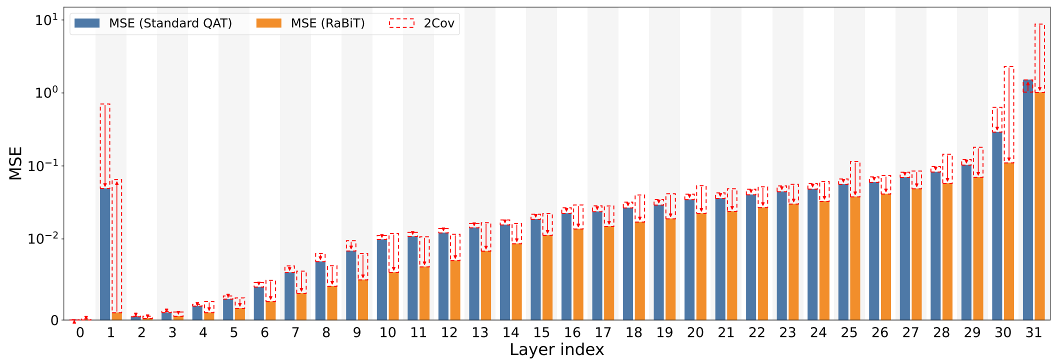


Figure 6: **Layer-wise MSE Decomposition in Llama2-7B's `down_proj` layers.** The bars compare the total Mean Squared Error (MSE) for Standard QAT (blue) and RaBiT (orange). The top of each bar represents the base error term (C), while the red-dashed component visualizes twice the covariance ($2 \times \text{Cov}$). RaBiT consistently generates a large negative covariance, which actively reduces the total MSE, demonstrating effective error cancellation. Notably, RaBiT also suppresses the extremely high MSE peak observed in the early layers of the Standard QAT baseline, indicating its robustness against layer sensitivity.

Second, and more strikingly, the analysis reveals RaBiT's ability to overcome a critical optimization challenge in extreme quantization: layer sensitivity. The Standard QAT baseline exhibits an exceptionally high MSE in the initial layers, a phenomenon consistent with the known sensitivity of early network layers to input distributions and quantization errors, as also observed in other LLMs by (Zhang et al., 2025; 2024). RaBiT, however, dramatically suppresses this MSE peak. This suggests its benefits extend beyond merely enforcing anti-correlation. The fact that RaBiT tames this instability indicates that our method may resolve a more fundamental bottleneck in MSE-based QAT that has historically hindered extreme quantization in conventional architectures. While we designed RaBiT to foster negative correlation, its success in stabilizing these sensitive layers points to a deeper robustness. A full investigation into how residual coupling imparts this stability is a compelling direction for future research.

A.7 Generated Samples

Sample 1: Newton

Prompt: Newton discovered the law of universal gravitation. Universal gravitation is

Generated Output

Newton discovered the law of universal gravitation. Universal gravitation is the force of gravity which draws objects towards the earth and other massive bodies. Gravity is the universal attraction between two objects and is proportional to the product of the masses of the two objects and inversely proportional to the square of the distance between them. Newton explained the force of gravity as being caused by the presence of matter in the universe. He stated that all matter attracted all other matter and that the tendency to move towards a massive body was proportional to the product of the masses of the objects. Newton developed the law of universal gravitation so that the attraction between two bodies could be calculated.

Sample 2: Adam Smith

Prompt: Adam Smith wrote The Wealth of Nations. The Wealth of Nations is

Generated Output

Adam Smith wrote The Wealth of Nations. The Wealth of Nations is the first book to outline the modern principles of capitalism and to make the case for it. Smith's work is widely considered to be the first comprehensive system of political economy and one of the most important works of political philosophy. The Wealth of Nations is the foundation of modern capitalism. The Wealth of Nations is a book by Adam Smith, first published in 1776, which describes the nature of economic systems. Adam Smith is regarded as the father of modern economics. The Wealth of Nations is widely considered to be the first comprehensive system of political economy and one of the most important works of political philosophy. It is also one of the most influential books on economics. Adam Smith is regarded as the father of modern economics.

Sample 3: DNA double helix

Prompt: Watson and Crick discovered the DNA double helix. The DNA double helix is

Generated Output

Watson and Crick discovered the DNA double helix. The DNA double helix is a structure found in the nucleus of every living cell. The double helix is the most fundamental structure in DNA. The structure is made of two single strands of DNA. The double helix was discovered in 1953 by James Watson and Francis Crick. In 1953, James Watson and Francis Crick discovered the double helix structure of DNA. The double helix structure is the most fundamental structure in DNA. The double helix structure is made of two single strands of DNA. The double helix is found in the nucleus of every living cell.

A.8 Algorithms

Algorithm 1 RaBiT Initialization: Calibrated Iterative Residual SVID

```
 1: **Require:** Pretrained weight $\mathbf{W}_{\mathrm{FP}}$, Number of paths $k$, Max iterations $T_{\max}$
 2: **Require:** Calibration stats $\mathbf{s}_{\mathrm{in}}, \mathbf{s}_{\mathrm{out}}$ and intensities $\alpha_{\mathrm{in}}, \alpha_{\mathrm{out}}$
 3: **Output:** Initialized scales $\{(\mathbf{g}_i, \mathbf{h}_i)\}_{i=1}^{k}$
 4: // Step 1: I/O Channel Importance-Calibrated Preconditioning
 5: Normalize: $\mathbf{s}_{\mathrm{in}} \leftarrow \mathbf{s}_{\mathrm{in}}/\max(\mathbf{s}_{\mathrm{in}})$,    $\mathbf{s}_{\mathrm{out}} \leftarrow \mathbf{s}_{\mathrm{out}}/\max(\mathbf{s}_{\mathrm{out}})$
 6: Precondition: $\mathbf{W}' \leftarrow \mathbf{s}_{\mathrm{out}}^{\alpha_{\mathrm{out}}} \odot \mathbf{W}_{\mathrm{FP}} \odot \mathbf{s}_{\mathrm{in}}^{\alpha_{\mathrm{in}}}$
 7: // Step 2: Iterative Residual SVID
 8: Initialize $\hat{\mathbf{W}}_i'^{(0)} \leftarrow \mathbf{0}$ for $i = 1, \ldots, k$
 9: **for** $t = 1$ to $T_{\max}$ **do**
10:     **for** $i = 1$ to $k$ **do**
11:         // Calculate target residual (Gauss-Seidel style update)
12:         $\mathbf{R}_i'^{(t)} \leftarrow \mathbf{W}' - \left(\sum_{j<i} \hat{\mathbf{W}}_j'^{(t)} + \sum_{j>i} \hat{\mathbf{W}}_j'^{(t-1)}\right)$
13:         // Apply SVID to find the best rank-1 approximation
14:         $(\mathbf{B}_i'^{(t)}, \mathbf{g}_i'^{(t)}, \mathbf{h}_i'^{(t)}) \leftarrow \mathrm{SVID}(\mathbf{R}_i'^{(t)})$
15:         $\hat{\mathbf{W}}_i'^{(t)} \leftarrow \mathbf{g}_i'^{(t)} \odot \mathbf{B}_i'^{(t)} \odot \mathbf{h}_i'^{(t)}$
16:     **end for**
17: **end for**
18: // Step 3: Map scales back to the original weight domain
19: **for** $i = 1$ to $k$ **do**
20:     $\mathbf{g}_i \leftarrow \mathbf{s}_{\mathrm{out}}^{-\alpha_{\mathrm{out}}} \odot \mathbf{g}_i'^{(T_{\max})}$
21:     $\mathbf{h}_i \leftarrow \mathbf{s}_{\mathrm{in}}^{-\alpha_{\mathrm{in}}} \odot \mathbf{h}_i'^{(T_{\max})}$
22: **end for**
```

Algorithm 2 RaBiT: Residual-Aware Binarization Training (One Step)

```
 1: **Parameters:** Shared full-precision weight $\mathbf{W}_{\mathrm{FP}}$; Scales $\{(\mathbf{g}_i, \mathbf{h}_i)\}_{i=1}^{k}$.
 2: **Input:** Minibatch Input $\mathbf{X}$, Targets $\mathbf{T}$.
 3: // 1. Forward Pass: On-the-fly Residual Coupling
 4: $\mathbf{R}_0 \leftarrow \mathbf{W}_{\mathrm{FP}}$.    // Initialize residual with the shared weight
 5: $\hat{\mathbf{W}}^{(k)} \leftarrow \mathbf{0}$.    // Effective weight for the entire layer
 6: **for** $i = 1$ to $k$ **do**
 7:     // Sequentially derive the i-th binary path
 8:     $\mathbf{B}_i \leftarrow \mathrm{sign}(\mathbf{R}_{i-1})$.
 9:     $\hat{\mathbf{W}}_i \leftarrow \mathbf{g}_i \odot \mathbf{B}_i \odot \mathbf{h}_i$.
10:     $\hat{\mathbf{W}}^{(k)} \leftarrow \hat{\mathbf{W}}^{(k)} + \hat{\mathbf{W}}_i$.
11:     // Update residual for the next path
12:     $\mathbf{R}_i \leftarrow \mathbf{R}_{i-1} - \hat{\mathbf{W}}_i$.
13: **end for**
14: $\mathbf{Y} \leftarrow \hat{\mathbf{W}}^{(k)}\mathbf{X}$.    // Compute layer output
15: Calculate Loss $\mathcal{L}(\mathbf{Y}, \mathbf{T})$.
16: // 2. Backward Pass
17: $\Delta \leftarrow \partial\mathcal{L}/\partial\mathbf{Y}$.    (Output gradient)
18: // Surrogate gradient for the shared weight $\mathbf{W}_{\mathrm{FP}}$
19: $\nabla_{\mathbf{W}_{\mathrm{FP}}} \leftarrow \Delta^{\top}\mathbf{X}$.
20: // Gradients for scales (treating $\mathbf{B}_i$ as constant)
21: **for** $i = 1$ to $k$ **do**
22:     Compute $\nabla_{\mathbf{g}_i}$ and $\nabla_{\mathbf{h}_i}$ using $\Delta, \mathbf{B}_i, \mathbf{X}$, and other scales.
23: **end for**
24: // 3. Parameter Update
25: Update $\{\mathbf{W}_{\mathrm{FP}}, (\mathbf{g}_i, \mathbf{h}_i)_{i=1}^{k}\}$ using an optimizer with the computed gradients.
```

A.9 LLM Usage

LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research.