RTAQ: REASONING TERMINATION-AWARE QUANTIZATION FOR MIXTURE-OF-EXPERTS ARCHITECTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Extreme low-bit post-training quantization is key to serving large-scale mixture-of-experts (MoE) language models, yet reasoning models quantized to around two bits often collapse on long-horizon reasoning tasks. We show that this collapse stems not from a loss of knowledge but from a failure to stop reasoning properly. When a 2-bit or near-1-bit reasoning model emits the end-of-thinking token (`</think>`), it terminates in every case we observe and answers nearly as accurately as its full-precision counterpart; most errors arise from reasoning traces that never close. We trace this termination decision to a small subset of routed experts (approximately 2% of all routed experts) that are consistently selected at the end-of-thinking position but are largely overlooked by standard sensitivity metrics. Building on this finding, we propose RTAQ (Reasoning-Termination-Aware Quantization), a weight-only quantization pipeline with three stages: (1) constructing a bank of activation-weighted, DP-optimal scalar codebooks for every expert and bit-width; (2) performing termination-aware bit allocation, which fixes termination experts at higher precision within the bit budget and assigns the remaining bits through knapsack optimization; and (3) applying grid-preserving recalibration, which reassigns codes using GPTQ-style error compensation and rescales expert outputs in under one second per expert. At an average bit-width of 2.0 for routed experts, RTAQ preserves over 95% of the full-precision performance on long-reasoning benchmarks such as AIME.

1 INTRODUCTION

Frontier language models have increasingly converged on the Mixture-of-Experts (MoE) Transformers, including GLM (Zeng et al., 2026), DeepSeek (Liu et al., 2024) and MiniMax (Chen et al., 2026). By activating only a few experts per token, MoE decouples model capacity from per-token compute, enabling models with hundreds of billions to trillions of parameters. However, expert weights still dominate memory and must remain available during inference. Indeed, more than 97% of the parameters in Qwen3.6-35B model reside in MoE blocks. Weight quantization is therefore particularly attractive for MoE models, and pushing routed-expert weights toward the 2-bit regime can substantially reduce their memory footprint.

However, aggressive quantization can cause a failure mode beyond accuracy degradation: reasoning models may fail to terminate properly. Instead of emitting `</think>` and proceeding to the final answer, they may continue reasoning excessively or fall into repetitive loops (Sui et al., 2025; Lotfi et al., 2026). This quantization-induced overthinking increases decoding latency and KV-cache usage in addition to degrading accuracy. Existing mixed-precision methods allocate bits using generic criteria such as reconstruction error or loss sensitivity (Dong et al., 2020; Shen et al., 2020; Hooper et al., 2025; Chitty-Venkata et al., 2025b), without explicitly accounting for reasoning termination.

Our key observation is that *reasoning termination is concentrated in a small subset of routed experts*. At positions immediately preceding `</think>`, Figure 2a shows that most experts are rarely selected, whereas a small subset is activated far more frequently than expected under uniform routing. We call these experts *termination-aware experts*. This observation suggests that preserving a very small, functionally important subset of experts at higher precision may protect reasoning termination without sacrificing the memory benefits of aggressive quantization.

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

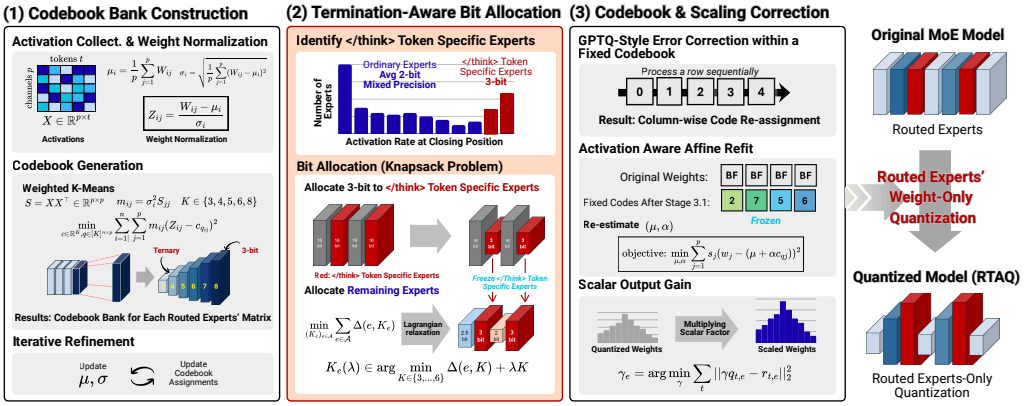


Figure 1: Overview of RTAQ (Reasoning Terminate-Aware Quantization).

Motivated by this finding, we propose RTAQ (**R**easoning **T**ermination-**A**ware **Q**uantization), a mixed-precision quantization framework for near-2-bit MoE models. RTAQ first constructs an activation-weighted codebook bank spanning multiple quantization levels for each routed expert. It then identifies termination-aware experts from their activation rates at closing positions and preserves them at higher precision (3 bits). For the remaining experts, RTAQ estimates loss-aware quantization sensitivity and allocates precision under a global bit budget through a Lagrangian relaxation of the resulting multiple-choice knapsack problem. Finally, it refines the selected quantized experts using GPTQ-style error compensation and a scalar output correction, without changing the allocated codebook sizes or storage format. An overview of RTAQ is illustrated in Figure 1.

We evaluate RTAQ on three sparse MoE models spanning 30B to 80B parameters and five reasoning and coding benchmarks. At only 2.016–2.025 bits per routed-expert weight on average, RTAQ achieves the highest average accuracy among all evaluated quantization methods on every model, outperforming the strongest competing method. Moreover, RTAQ keeps generation lengths close to those of the original BF16 models, demonstrating that preserving termination-aware experts helps mitigate the excessive reasoning induced by aggressive quantization.

Our main contributions are:

- We identify a small subset of routed experts that are activated at unusually high rates immediately before `</think>`. We call them *termination-aware experts* and use them to preserve reasoning termination under quantization.
- We introduce RTAQ, a termination-aware mixed-precision framework combining an activation-weighted codebook bank, high-precision protection of termination-aware experts, loss-aware sensitivity, Lagrangian bit allocation, and fixed-codebook GPTQ-style correction.
- Across three MoE models and five benchmarks, RTAQ achieves the highest accuracy at the lowest average bit-width among evaluated methods. With only 2.02 bits per routed-expert weight, it retains 85.81% of average accuracy on the Qwen models while BF16 model achieves 87.85%. We maintain near-BF16 generation lengths and mitigating quantization-induced overthinking.

1.1 Related Work

Quantization of Mixture-of-Experts Models. Recent work reduces the memory footprint of Mixture-of-Experts (MoE) models through low-bit quantization. Early methods (Kim et al., 2023; Frantar & Alistarh, 2023) focused on extreme low-bit compression of expert weights (*e.g.*, sub-1-bit, 2-bit), but uniform bit-widths ignore differences in quantization sensitivity (Li et al., 2024). Subsequent works (Huang et al., 2025; Duanmu et al., 2025; Chitty-Venkata et al., 2025a; Chowdhury et al., 2026) therefore introduce mixed-precision quantization, assigning bit-widths according to expert importance. They use signals (*e.g.*, Hessian-based sensitivity and reconstruction error) to preserve important experts at higher precision. More recent work extends this idea to model-wide bit-width optimization across experts (Deng et al., 2026). Despite these advances, existing methods rely on generic quantization criteria or aggregate task performance, without explicitly considering each expert's role in reasoning termination.

Low-Bit Quantization for Reasoning Models. Several studies have shown that low-bit quantization can substantially degrade LLM reasoning (Liu et al., 2025; Zhang et al., 2026). Quantized models may generate unnecessarily long reasoning traces (Lian et al., 2026), exhibit overthinking and delayed commitment (Lotfi et al., 2026), and show repetitive loops or termination failures at extreme low bit-widths (Alimaskina et al., 2026). Recent approaches leverage self-generated reasoning traces for calibration (Wang et al., 2026) and reasoning-aware QAT to recover degraded reasoning capability (Okoshi et al., 2026). These efforts mainly target model-wide quantization error reduction or the recovery of reasoning capability. However, they do not explicitly consider which model components are critical to reasoning termination. In contrast, we quantize a small subset of termination-aware experts to 3 bits and the remaining experts to an average of 2 bits.

2 PRELIMINARIES

We consider a Transformer in which every block contains a sparse Mixture-of-Experts (MoE) layer (Shazeer et al., 2017; Fedus et al., 2022) with a shared expert. In each MoE layer, a router selects a small subset of experts for each token and combines their outputs using routing weights.

Quantization Representation. We approximate a pretrained weight matrix $W \in \mathbb{R}^{n\times p}$ by

$$\widehat{W}_{ij} = \mu_i + \sigma_i c_{q_{ij}}, \qquad q_{ij} \in \{1, \dots, K\}, \tag{1}$$

where $c \in \mathbb{R}^K$ contains the quantization levels shared across rows, q_{ij} selects a codebook entry, and $\mu, \sigma \in \mathbb{R}^{n\times 1}$ are row-wise offsets and scales. We use $\log_2 K$ logical bits per weight to describe code precision; actual storage also depends on index packing and the storage of the codebook and row-wise affine parameters.

Given calibration inputs $X \in \mathbb{R}^{p\times t}$, layer-wise quantization minimizes the output reconstruction error (Nagel et al., 2020; Frantar et al., 2023)

$$\mathcal{E}(\widehat{W}) = \left\|(W - \widehat{W})X\right\|_F^2 = \mathrm{tr}\Big((W - \widehat{W})S(W - \widehat{W})^\top\Big), \tag{2}$$

where $S = XX^\top$ is the unnormalized second-moment matrix. Under a diagonal approximation to S, the objective reduces to $\sum_{i,j} S_{jj}(W_{ij} - \widehat{W}_{ij})^2$, weighting each squared error by the activation energy of its input dimension. The off-diagonal entries capture interactions between input dimensions, which GPTQ-style error compensation uses to adjust the remaining weights after quantization.

Loss Sensitivity for Mixed-Precision Quantization. Mixed-precision methods allocate precision under a budget according to the increase in task loss $\mathcal{L}$ caused by quantization (Dong et al., 2019; 2020). Here, $\mathcal{L}$ is the model's average next-token prediction loss, so sensitivity measures the effect on the model's predictions beyond local reconstruction error. If quantizing a block changes its vectorized output h to $h + \delta$, a second-order Taylor expansion gives

$$\Delta\mathcal{L} \approx g^\top\delta + \frac{1}{2}\delta^\top H\delta, \tag{3}$$

where $g = \nabla_h\mathcal{L}$ and $H = \nabla_h^2\mathcal{L}$ are evaluated at the original BF16 model. The quadratic term is commonly used as a sensitivity surrogate (Dong et al., 2019), with curvature approximated using gradient outer products as in empirical-Fisher methods (Martens, 2020; Kunstner et al., 2019).

Reasoning Termination. Recent language models that support *thinking* mode (OpenAI, 2024; Guo et al., 2025; Yang et al., 2025b) can generate an explicit chain of thought before producing a final answer. For example, Qwen3 models (Yang et al., 2025a) enclose the reasoning trace between `<think>` and `</think>`, then generate the final answer followed by an end-of-sequence token. A response *terminates* if it emits the end-of-sequence token within the generation budget and is *truncated* otherwise.

Low-bit quantization can degrade reasoning performance (Liu et al., 2025). At extreme bit-widths, we focus on *reasoning non-termination*, in which the reasoning trace loops or drifts until the generation budget is exhausted without emitting `</think>` or producing a final answer. This failure is related to *overthinking* observed in full-precision reasoning models, where reasoning continues beyond what the problem requires even when the response eventually terminates (Chen et al., 2024; Sui et al., 2025).

3 RTAQ: REASONING TERMINATION-AWARE QUANTIZATION FOR MOE

We introduce RTAQ, a mixed-precision quantization framework for routed experts that preserves higher precision for termination-aware experts. First, we construct a *codebook bank* containing multiple quantization levels for each expert. Second, we allocate adaptive bit-widths across experts under a target budget while protecting termination-aware experts. Finally, we refine the selected quantized experts without changing their codebooks.

3.1 STAGE I: ACTIVATION-WEIGHTED CODEBOOK BANK

We first construct a *codebook bank* for each routed-expert weight, covering multiple quantization levels (i.e., codebook size) $K \in \{3, \dots, 8\}$. The bank is constructed once offline and subsequently reused for different target bit budgets. Given a matrix $W \in \mathbb{R}^{n \times p}$, we first normalize each row as

$$Z_{ij} = \frac{W_{ij} - \mu_i}{\sigma_i}, \qquad \mu_i = \frac{1}{p}\sum_{j=1}^{p} W_{ij}, \qquad \sigma_i = \sqrt{\frac{1}{p}\sum_{j=1}^{p}\left(W_{ij} - \mu_i\right)^2}, \tag{4}$$

where $\mu, \sigma \in \mathbb{R}^{n \times 1}$ are the row-wise means and the standard deviations, respectively. These vectors incur only a small quantization overhead, as each (μ_i, σ_i) is shared across all entries in row i.

For each quantization level $K \in \{3, \dots, 8\}$, we obtain a codebook $c \in \mathbb{R}^K$ by applying weighted K-means clustering to vectorized Z. Rather than weighting all coordinates equally, we utilize their expected contribution to the layer output through an *activation-weighted mass*. Formally, let $X \in \mathbb{R}^{p \times t}$ denote calibration activations and denote $S = XX^\top \in \mathbb{R}^{p \times p}$. For each normalized weight Z_{ij} for $i \in [n], j \in [p]$, we assign the mass $m_{ij} = \sigma_i^2 S_{jj}$ where S_{jj} measures the activation energy associated with input dimension j. The codebook and assignments are obtained by minimizing

$$\min_{c \in \mathbb{R}^K, q \in [K]^{n \times p}} \sum_{i=1}^{n}\sum_{j=1}^{p} m_{ij}\left(Z_{ij} - c_{q_{ij}}\right)^2. \tag{5}$$

We then refine the affine parameters μ and σ to further reduce the reconstruction error. Starting from the row-wise statistics in Equation (4) and the weighted K-means solution, we perform alternating optimization: with μ fixed, we optimize σ; with σ fixed, we optimize μ. After each update, we recompute the normalized weights and update their codeword assignments accordingly. We repeat these steps 5 times. After this offline stage, each routed expert is equipped with a codebook bank spanning $\log_2 3$ to 3 bits per weight, from which a codebook can later be selected according to the desired expert-wise bit allocation. Additional details are provided in Section C.

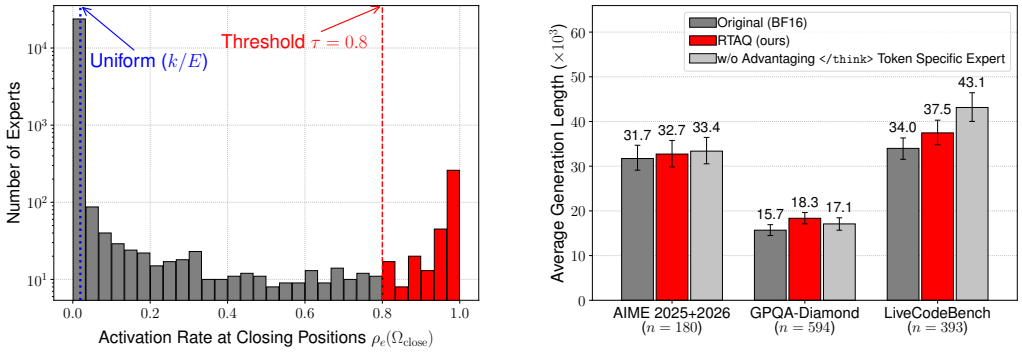
3.2 STAGE II: TERMINATION-AWARE MIXED-PRECISION ALLOCATION

Heavily quantized models often fail to terminate their reasoning at the appropriate time, resulting in excessive reasoning (overthinking) or repetitive loops (Sui et al., 2025; Lotfi et al., 2026). We find that in MoE models, the prediction of the end-of-reasoning token `</think>` is routed to a small and consistent subset of experts, which we call *termination-aware experts*. We first identify these experts via empirical activation rates and protect them from aggressive quantization and then allocate precision to the remaining experts under a global bit budget.

Reasoning Termination-Aware Experts. Consider a MoE model with L layers and E routed experts per layer, where each router selects k experts for every token. Let $\mathcal{E}$ denote the set of all routed experts across all layers, $|\mathcal{E}| = LE$. We collect N sequences generated by the model, $\{x^{(i)}\}_{i=1}^{N}$ where $x^{(i)} = (x_1^{(i)}, \dots, x_{T_i}^{(i)})$ consists of the i-th prompt followed by the generated response. We keep only sequences that contain the closing token `</think>`. Let $\mathcal{T}_{i,t} \subseteq \mathcal{E}$ denote the set of experts activated at position t of $x^{(i)}$, collected over all layers.

We call position t a *closing position* if the next token is `</think>`, i.e., $x_{t+1}^{(i)} =$ `</think>`. Thus, the closing position represents the state at which the model predicts the termination of its reasoning process. Let $c_i \in [T_i - 1]$ denote the closing position of sequence i, and define

$$\Omega_{\text{close}} = \{(i, c_i) : i \in [N]\}. \tag{6}$$



(a) (b)

Figure 2: **Emergence of termination-aware experts.** (a) Distribution of routed-expert activation rates at closing positions (i.e., the token immediately preceding </think>) for Qwen3-Next-80B-A3B on GPQA-Diamond. (b) Average generation length with and without termination-aware expert pinning. Preserving termination-aware experts at higher precision keeps RTAQ close to the BF16 generation length, whereas ignoring them leads to overthinking with substantially longer outputs.

For any set of token positions Ω, we define the *activation rate* of expert $e \in \mathcal{E}$ as

$$\rho_e(\Omega) = \frac{1}{|\Omega|} \sum_{(i,t)\in\Omega} \mathbb{I}[e \in \mathcal{T}_{i,t}] \in [0,1], \tag{7}$$

where $\mathbb{I}[\cdot]$ denotes the indicator function. Given a threshold $\tau \in (0,1)$, we define the set of *termination-aware experts* as

$$\mathcal{P} := \{e \in \mathcal{E} \ : \ \rho_e(\Omega_{\text{close}}) \geq \tau\}. \tag{8}$$

That is, an expert is termination-aware if it is activated on at least a fraction τ of the closing positions.

Figure 2 demonstrates the empirical evidence of termination-aware experts and statistics. Figure 2a shows that activation rates at closing positions are highly skewed across routed experts. Most experts are rarely selected, while a small subset is activated far more frequently than expected under uniform routing, where the expected rate is k/E. This motivates preserving these *termination-aware experts* during quantization. In practice, we set $\tau = 0.8$. Figure 2b shows that removing a set of termination-aware experts $\mathcal{P}$ consistently increases generation length, whereas RTAQ keeping $\mathcal{P}$ higher bit (3-bit) remains closer to the original BF16 model. This confirms that preserving these experts is important for mitigating quantization-induced overthinking.

We therefore assign these experts in $\mathcal{P}$ to a higher bit (i.e., $K = 8$ levels) and allocate the remaining bit budget adaptively across the other experts in $\mathcal{E} \setminus \mathcal{P}$.

Loss-Aware Expert Sensitivity. To allocate adaptive bit-widths across experts, we quantify the sensitivity of expert e to K-level quantization through the loss-aware perturbation of its MoE block. Second-order sensitivity measures of this form have been widely used to guide mixed-precision quantization (Dong et al., 2020; Shen et al., 2020; Hooper et al., 2025; Chitty-Venkata et al., 2025b; NVIDIA). Given a calibration sequence $x = (x_1, \ldots, x_T)$, let $\mathcal{L}(x) = \frac{1}{T}\sum_{t=1}^{T} \ell_t$ denote the average next-token prediction loss, and let h_t be the output of the MoE block containing expert e at token position t. We define

$$g_t = \frac{\partial \mathcal{L}(x)}{\partial h_t}, \qquad \delta_t(e,K) = w_{t,e}\left(f_{e,K}(u_t) - f_e(u_t)\right), \tag{9}$$

where u_t is the input to the MoE layer, $w_{t,e}$ is the routing weight assigned to expert e, and f_e and $f_{e,K}$ denote the original and K-level quantized expert functions, respectively. We set $w_{t,e} = 0$ when expert e is not selected. Thus, $\delta_t(e,K)$ represents the local perturbation to the MoE output induced by quantizing only expert e. Using a token-wise block-diagonal Fisher approximation to the loss

curvature, we define the sensitivity of the (e, K) pair on sequence x as

$$\Delta(e, K; x) = \frac{T}{2} \sum_{t=1}^{T} \langle g_t, \delta_t(e, K) \rangle^2 = \frac{T}{2} \sum_{t=1}^{T} \delta_t(e, K)^\top [g_t g_t^\top] \delta_t(e, K) \tag{10}$$

The factor T accounts for the fact that g_t is computed from the sequence-averaged loss $\mathcal{L}(x)$. Equivalently, each curvature block is approximated as $H_t \approx T g_t g_t^\top$.

Intuitively, $\Delta(e, K; x)$ is large when quantizing expert e induces a perturbation along directions to which the prediction loss is highly sensitive, while ignoring tokens that are not routed to expert e, i.e., $w_{t,e} = 0$. Given a calibration set $\{x^{(i)}\}_{i=1}^N$, we estimate the sensitivity of each expert–level pair by averaging across sequences: $\Delta(e, K) = \frac{1}{N} \sum_{i=1}^N \Delta\left(e, K; x^{(i)}\right)$. Additional details on the curvature approximation are provided in Section D.1.

Mixed-Precision Allocation via Lagrangian Relaxation. Let $\mathcal{A} = \mathcal{E} \setminus \mathcal{P}$ denote the set of non-termination-aware experts. Given a target average bit-width b_{target}, we reserve the maximum codebook size $K_{\max} = 8$ for termination-aware experts in $\mathcal{P}$. The remaining bit budget for experts in $\mathcal{A}$ is therefore $B = |\mathcal{E}| b_{\text{target}} - |\mathcal{P}| \log_2 K_{\max}$. The mixed-precision allocation problem is then formulated as

$$\min_{(K_e)_{e \in \mathcal{A}}} \sum_{e \in \mathcal{A}} \Delta(e, K_e) \qquad \text{s.t.} \qquad \sum_{e \in \mathcal{A}} \log_2 K_e \le B, \quad K_e \in \{3, \dots, 6\}. \tag{11}$$

This problem is a multiple-choice knapsack problem (Sinha & Zoltners, 1979) which is NP-hard in general. A naïve approach is to greedily increase the precision of the expert that yields the largest sensitivity reduction per additional bit. However, it considers only local transitions between neighboring configurations and can be suboptimal when the sensitivity–precision tradeoff is non-convex. We instead use a Lagrangian relaxation that compares all candidate precisions for each expert simultaneously (Everett, 1963; Shoham & Gersho, 1988). Full details are provided in Section D.2.

3.3 Stage III: Codebook and Scaling Correction

After mixed-precision allocation, we further refine each selected quantized expert without changing its quantization level, codebook, or storage format. Specifically, we improve the discrete codeword assignments using GPTQ-style error compensation, update the row-wise affine parameters and apply a scalar output correction to compensate for residual expert-level distortion. Importantly, the normalized codebook selected in the previous stage remains fixed throughout this stage.

GPTQ-Style Error Correction within a Fixed Codebook. We first refine the codeword assignments using an OBS/GPTQ-style sequential compensation procedure. Unlike standard GPTQ, which quantizes weights onto a predefined uniform grid, we quantize each compensated weight to its nearest entry in the selected codebook. For a current compensated weight value w_j, row-wise affine parameters (μ, σ), and a fixed normalized codebook $c \in \mathbb{R}^K$, we assign

$$q_j = \arg\min_{q \in [K]} |w_j - (\mu + \sigma c_q)|, \tag{12}$$

and reconstruct it as $\widehat{w}_j = \mu + \sigma c_{q_j}$, $\varepsilon_j = w_j - \widehat{w}_j$. As in GPTQ, the quantization error ε_j is propagated to the remaining unprocessed weights using an inverse-curvature-based compensation update. Crucially, Equation (12) is applied to the *current compensated value* of w_j, rather than its original value, so each assignment accounts for errors introduced by previously quantized weights.

This procedure does not construct a new codebook or alter the allocated quantization level. It only updates the discrete assignments within the fixed normalized codebook. Therefore, the corrected expert retains the same codebook size and storage format, while benefiting from GPTQ-style error compensation.

Scalar Output Gain. Even after weight-level correction, residual error can remain in the expert output. In particular, quantization errors in the gate and up projections pass through the nonlinear SiLU gating and subsequently interact with the quantized down projection, which can introduce a systematic bias in the output magnitude. We correct this effect using a single scalar gain per expert.

Table 1: Performance comparison of various quantization methods across 3 models on math, reasoning and coding benchmarks. All reported results are accuracies (%; higher is better). "Bits" denotes the average number of bits per weight in the routed experts. Baseline denotes the original unquantized model in BF16 precision. Bold indicates the best result for each model and benchmark.

Method	Bits	GSM8K	AIME25	AIME26	GPQA-Diamond Strict.	GPQA-Diamond Flex.	LCB	Average
Qwen3.6-35B-A3B								
Baseline	16	96.8	91.1	91.1	82.3	83.0	77.6	87.85
GPTQ (Frantar et al., 2022)	2.250	95.7	80.0	81.1	71.2	72.9	50.1	75.79
MC-MoE (Huang et al., 2025)	2.358	96.3	88.9	85.6	71.9	72.6	**72.3**	83.07
GSQ (Dadgarnia et al., 2026)	2.125	96.0	84.4	86.7	**79.8**	**80.6**	66.4	82.74
TileQ (Gu et al., 2026)	2.320	95.2	82.2	86.7	73.7	71.7	49.4	77.24
RTAQ (Ours)	2.022	**96.6**	**90.0**	**93.3**	78.6	79.3	70.2	**85.81**
Nemotron-3-Nano-30B-A3B								
Baseline	16	96.1	87.8	90.0	72.7	73.1	66.2	82.60
GPTQ (Frantar et al., 2022)	2.250	**94.7**	**85.6**	80.0	58.9	59.4	51.9	74.27
MC-MoE (Huang et al., 2025)	2.358	94.2	81.1	84.4	63.8	63.8	58.0	76.30
GSQ (Dadgarnia et al., 2026)	2.125	93.6	82.2	82.2	38.2	37.7	60.1	71.21
TileQ (Gu et al., 2026)	2.301	94.0	81.1	74.4	60.6	60.4	52.2	72.44
RTAQ (Ours)	2.025	94.3	81.1	**85.6**	**67.2**	**67.0**	**60.3**	**77.68**
Qwen3-Next-80B-A3B-Thinking								
Baseline	16	96.6	85.6	88.9	75.6	75.3	68.2	82.95
GPTQ (Frantar et al., 2022)	2.250	96.1	78.9	87.8	65.2	63.3	58.3	77.07
MC-MoE (Huang et al., 2025)	2.358	96.1	**84.4**	**91.1**	59.6	58.8	57.8	77.72
GSQ (Dadgarnia et al., 2026)	2.125	**96.4**	83.3	84.4	62.3	58.2	56.0	76.07
TileQ (Gu et al., 2026)	2.307	96.1	80.0	87.8	68.2	68.7	61.6	78.79
RTAQ (Ours)	2.016	**96.4**	**84.4**	88.9	**72.2**	**70.5**	**64.1**	**81.03**

For a calibration token t, let $q_{t,e} = w_{t,e} f_{e,K}(x_t)$, $r_{t,e} = w_{t,e} f_e(x_t)$ where $f_{e,K}$ and f_e denote the quantized expert and the original BF16 expert, respectively, and $w_{t,e}$ is the routing weight. We choose the gain γ_e that best aligns the quantized expert output with its BF16 counterpart:

$$\gamma_e = \arg\min_{\gamma \in \mathbb{R}} \sum_t |\gamma q_{t,e} - r_{t,e}|_2^2 = \frac{\sum_t \langle q_{t,e}, r_{t,e} \rangle}{\sum_t |q_{t,e}|_2^2}. \tag{13}$$

This scalar corrects systematic output shrinkage or amplification that is not captured by local weight reconstruction alone. We fold γ_e into the affine scale and offset of the expert's down projection, so it introduces no additional inference operation and requires no additional per-weight storage. Importantly, neither the discrete codes nor the codebook are modified. Further details are provided in Section E.1.

4 Experiments

Setup. We compare RTAQ against various post-training quantization baselines including GPTQ (Frantar et al., 2022) as a standard weight-only quantization method, MC-MOE (Huang et al., 2025) and TileQ (Gu et al., 2026) as MoE-specific quantization methods, and GSQ (Dadgarnia et al., 2026), a recent low-bit scalar quantization method. We evaluate these methods on three models spanning 30B to 80B parameters: Qwen3.6-35B-A3B, Nemotron-3-Nano-30B-A3B and Qwen3-Next-80B-A3B-Thinking. All models adopt sparse MoE architectures, with more than 97% of their parameters residing in the routed experts. We therefore quantize only the routed-expert parameters and choose method-specific hyperparameters such that each method uses approximately 2 bits per routed expert parameter on average.

We evaluate the resulting quantized models on five benchmarks: GSM8K (Cobbe et al., 2021), AIME25/26 (Zhang & Math-AI, 2025; 2026), GPQA-Diamond (Rein et al., 2023) and Live-

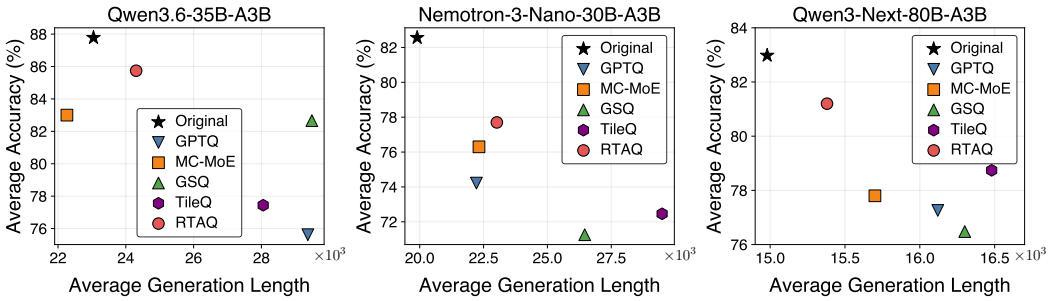


Figure 3: Accuracy–length trade-off across three models. Each point reports the average accuracy and generation length across five benchmarks. The original BF16 model (black, star) is included as a reference. Higher accuracy and shorter generation length are preferred (upper left). RTAQ (red, circle) consistently provides a favorable trade-off, substantially preserving the accuracy of the original model while maintaining generation lengths close to the original.

CodeBench (Jain et al., 2025), spanning mathematical reasoning, scientific reasoning, and code generation. For GPQA-Diamond, we report results under two evaluation criteria, strict-match and flexible-extract. All experiments were conducted on NVIDIA RTX PRO 6000 GPUs (with 96 GB VRAM), using two GPUs for 30B-parameter models and four GPUs for the 80B-parameter model.

4.1 Accuracy Preservation under Near-2-Bit Quantization

Table 1 summarizes accuracy on five benchmarks across three models under a near-2-bit routed-expert budget. Averaged over the benchmarks, RTAQ achieves the highest accuracy among all quantized methods on every model while using the fewest bits per routed-expert weight. Compared with the strongest baseline, MC-MoE, RTAQ improves average accuracy by 2.74, 1.38, and 3.31 points on Qwen3.6-35B-A3B, Nemotron-3-Nano-30B-A3B, and Qwen3-Next-80B-A3B-Thinking, respectively. It also remains close to BF16, retaining 97.7% of the original average accuracy on both Qwen models. Moreover, we found that RTAQ is substantially more robust across individual model–benchmark pairs. While competing methods suffer large drops on challenging tasks such as GPQA-Diamond and LiveCodeBench, the largest BF16-relative degradation of RTAQ is only 7.4 points, compared with 16.5–35.4 points for the baselines. These results show that RTAQ preserves accuracy more consistently in the near-2-bit regime.

4.2 Accuracy–Generation Length Trade-off

Low-bit quantization can also alter the reasoning behavior of MoE models, often causing excessively long reasoning traces or failure to emit `</think>` at an appropriate point. Figure 3 summarizes this quantization-induced overthinking through the accuracy–generation length trade-off, averaged over the five benchmarks. Across all three models, RTAQ better preserves BF16 accuracy while keeping generation length close to the original model. For example, on Qwen3-Next-80B-A3B-Thinking, RTAQ achieves 81.2% average accuracy versus 76.5–77.8% for competing methods, with an average generation length of 15.4K tokens compared with 15.0K for BF16. These results indicate that RTAQ mitigates both accuracy degradation and quantization-induced overthinking.

5 Ablation Studies

5.1 Bit Allocation Across Models

Figure 4 shows the resulting bit allocation. Although the knapsack algorithm assigns each expert its own level, it consistently places the largest share at the lowest one, i.e., $K{=}3$ (1.58 bits): 50.7%, 39.4% and 50.6% of the routed experts of Qwen3.6-35B-A3B, Nemotron-3-Nano-30B-A3B and Qwen3-Next-80B-A3B, respectively. The next most common level on both Qwen models is $K{=}6$ (31.2% and 32.0%). Levels above this rarely provide meaningful benefits. On Qwen3.6-35B-A3B, allowing up to $K{=}9$ lowers the predicted cost but raises the held-out NLL (0.4707 vs. 0.4672). The

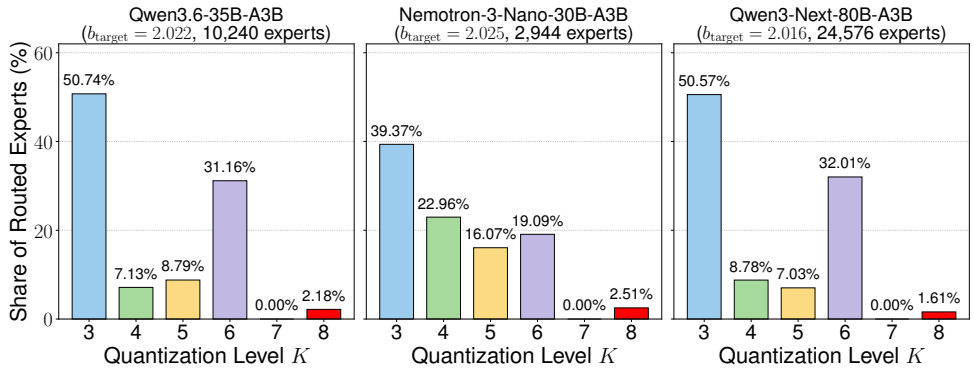
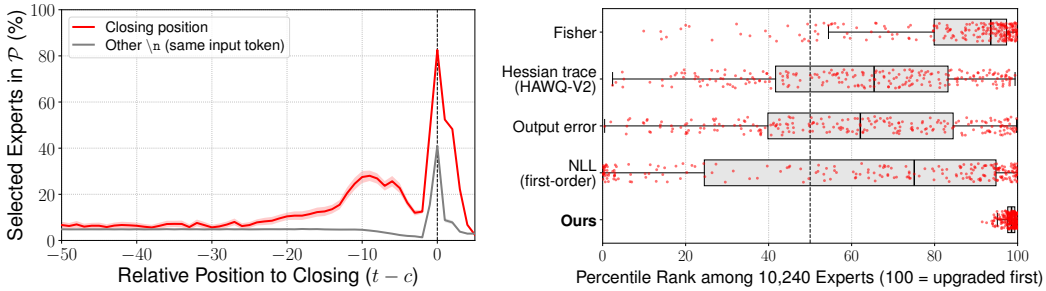


Figure 4: Percentage of routed experts assigned to each quantization level $K \in \{3, 4, 5, 6, 8\}$ by RTAQ. The red bars indicate the termination-aware experts, which are fixed at K=8.



(a) (b)

Figure 5: `</think>`-token-specific experts $\mathcal{P}$ in Qwen3.6-35B-A3B. (a) Percentage of the experts in $\mathcal{P}$ selected by the router at each token. Red: tokens just before the model ends its reasoning with `</think>` (0 on the x-axis). Grey: tokens around ordinary line breaks in the middle of the reasoning, shown for comparison. Most of $\mathcal{P}$ is selected only at the end of reasoning (373 responses from AIME 2025+2026 and GPQA-Diamond; shading: 95% CI). (b) How important each sensitivity criterion considers the experts in $\mathcal{P}$, as a rank among all 10,240 routed experts (100 = most important). Conventional criteria rank many of them in the middle or lower half, whereas Ours ranks all of them near the top. Dashed line: a random expert.

termination-aware experts are the exception. Held at K=8 outside the budget, they make up only 2.18%, 2.51% and 1.61% of the routed experts, yet, as Section 5.2 shows, they must keep this higher precision.

5.2 Termination-Aware Experts and Conventional Sensitivity Criteria

To verify that the `</think>`-token-specific experts $\mathcal{P}$ form a distinct set, we measure on Qwen3.6-35B-A3B the share of $\mathcal{P}$ selected at each position of a response, aligned to its closing position. As shown in Figure 5a, about 83% of $\mathcal{P}$ is selected at the closing position, against 5–7% during reasoning and 41% at other newlines, the input token at every closing position. Their activation thus follows the decision to stop rather than the input token.

We then examine whether sensitivity criteria used in mixed-precision quantization would protect these experts. As shown in Figure 5b, the Hessian trace, output error and NLL rank $\mathcal{P}$ near the middle (medians 62–75), and even the Fisher, which ranks them higher (median 94), leaves 39% outside its top 10%. Averaged over ordinary tokens, these criteria see $\mathcal{P}$ only at the few positions where it is decisive, and so would leave many of these experts at low bit-widths. RTAQ instead identifies $\mathcal{P}$ directly from routing statistics and holds it at K=8.

AI USE STATEMENT

We used generative AI tools to assist with figure and table editing, and routine coding. AI-assisted code was reviewed and tested by the authors, and all reported results were independently produced and verified by the authors. All AI-assisted text and figures were reviewed against the underlying experiments and data. The authors take full responsibility for the content of this paper.

REFERENCES

Ekaterina Alimaskina, Darya Rudas, Denis Shveykin, Gleb Molodtsov, Pavel Vasiliev, and Aleksandr Beznosikov. Extreme low-bit inference in reasoning models: Failure modes and targeted recovery. *arXiv preprint arXiv:2606.02011*, 2026. 3

Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009. 15

Richard Bellman. A note on cluster analysis and dynamic programming. *Management Science*, 19 (11):1328–1329, 1973. 15

Aili Chen, Aonian Li, Baichuan Zhou, Bangwei Gong, Binyang Jiang, Boji Dan, Changhao Zhang, Changqing Yu, Chao Wang, Cheng Ma, et al. The minimax-m2 series: Mini activations unleashing max real-world intelligence. *arXiv preprint arXiv:2605.26494*, 2026. 1

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? On the overthinking of o1-like LLMs. *arXiv preprint arXiv:2412.21187*, 2024. 3

Krishna Teja Chitty-Venkata, Jie Ye, and Murali Emani. Mopeq: Mixture of mixed precision quantized experts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 4082–4091, October 2025a. 2

Krishna Teja Chitty-Venkata, Jie Ye, and Murali Emani. Mopeq: Mixture of mixed precision quantized experts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 4082–4091, 2025b. 1, 5

Mohammed Nowaz Rabbani Chowdhury, Kaoutar El Maghraoui, Hsinyu Tsai, Naigang Wang, Geoffrey W. Burr, Liu Liu, and Meng Wang. Efficient quantization of mixture-of-experts with theoretical generalization guarantees. *arXiv preprint arXiv:2604.06515*, 2026. 2

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 7

Alireza Dadgarnia, Soroush Tabesh, Mahdi Nikdan, Michael Helcig, Eldar Kurtic, Maximilian Kleinegger, and Dan Alistarh. Gsq: Highly-accurate low-precision scalar quantization for llms via gumbel-softmax sampling. *arXiv preprint arXiv:2604.18556*, 2026. 7

Jianing Deng, Song Wang, Dongwei Wang, Zijie Liu, Tianlong Chen, Huanrui Yang, and Jingtong Hu. GEMQ: Global expert-level mixed-precision quantization for moe LLMs. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=5Q4ts0pr9T. 2

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: Hessian AWare quantization of neural networks with mixed-precision. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 3, 17

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 3, 5, 17

Haojie Duanmu, Xiuhong Li, Zhihang Yuan, Size Zheng, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. Mxmoe: Mixed-precision quantization for moe with accuracy and performance co-design. In *International Conference on Machine Learning*, 2025. 2

Hugh Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3):399–417, 1963. 6, 18

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. 3

Walter D. Fisher. On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798, 1958. 15

Elias Frantar and Dan Alistarh. Qmoe: Practical sub-1-bit compression of trillion-parameter models. *arXiv preprint arXiv:2310.16795*, 2023. 2

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022. 7

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023. 3

Allan Grønlund, Kasper Green Larsen, Alexander Mathiasen, Jesper Sindahl Nielsen, Stefan Schneider, and Mingzhou Song. Fast exact k-means, k-medians and Bregman divergence clustering in 1D. *arXiv preprint arXiv:1701.07204*, 2017. 15

Hongyaoxing Gu, Xinzhe Chen, Lijuan Hu, and Fangfang Liu. Tileq: Efficient low-rank quantization of mixture-of-experts with 2d tiling. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ReSBUj2514. 7

Daya Guo et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 3

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016. 22

Coleman Hooper, Charbel Sakr, Ben Keller, Rangharajan Venkatesan, Kurt Keutzer, Sophia Shao, and Brucek Khailany. Fgmp: fine-grained mixed-precision weight and activation quantization for hardware-accelerated llm inference. *arXiv preprint arXiv:2504.14152*, 2025. 1, 5

Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and XIAOJUAN QI. Mixture compressor for mixture-of-experts LLMs gains more. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=hheFYjOsWO. 2, 7

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025. 8

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *International Conference on Machine Learning (ICML)*, 2024. 22

Young Jin Kim, Raffy Fahim, and Hany Hassan Awadalla. Mixture of quantized experts (moqe): Complementary effect of low-bit quantization and robustness. *arXiv preprint arXiv:2310.02410*, 2023. 2

Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 3, 17

Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016. 16, 22

Pingzhi Li, Xiaolong Jin, Yu Cheng, and Tianlong Chen. Examining post-training quantization for mixture-of-experts: A benchmark, 2024. URL https://openreview.net/forum?id=sMwYn2lZjO. 2

Xinyu Lian, Walid Krichene, Beichen Huang, Masahiro Tanaka, Olatunji Ruwase, Li Zhang, and Minjia Zhang. Quantization inflates reasoning: Token inflation as a hidden cost of low-bit reasoning models. *arXiv preprint arXiv:2606.25519*, 2026. 3

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. 1

Ruikang Liu, Yuxuan Sun, Manyi Zhang, Haoli Bai, Xianzhi Yu, Tiezheng YU, Chun Yuan, and Lu Hou. Quantization hurts reasoning? an empirical study on quantized reasoning models. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=BM192Ps5Nv. 3

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28 (2):129–137, 1982. 15

Sanae Lotfi, Polina Kirichenko, Steven Li, and Zechun Liu. Quantized reasoning models think they need to think longer, but they do not. *arXiv preprint arXiv:2606.00206*, 2026. 1, 3, 4

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit LLMs: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024. 22

Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is NP-hard. *Theoretical Computer Science*, 442:13–21, 2012. 15

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. 3, 17

Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In *International Conference on Machine Learning (ICML)*, 2020. 3

Model Optimizer Team NVIDIA. Autoquantize: A fast automatic mixed-precision assignment. URL https://nvidia.github.io/Model-Optimizer/announcements/autoquantize.html. 5

Yasuyuki Okoshi, Hikari Otsuka, Daichi Fujiki, and Masato Motomura. Towards quantization-aware training for ultra-low-bit reasoning LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Azsd2qyK6C. 3

OpenAI. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. 3

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023. 7

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017. 3

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 8815–8821, 2020. 1, 5

Yair Shoham and Allen Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(9):1445–1453, 1988. 6, 18

Prabhakant Sinha and Andris A. Zoltners. The multiple-choice knapsack problem. *Operations Research*, 27(3):503–515, 1979. 6

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025. 1, 3, 4

Haizhou Wang and Mingzhou Song. Ckmeans.1d.dp: Optimal k-means clustering in one dimension by dynamic programming. *The R Journal*, 3(2):29–33, 2011. 15

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-aware automated quantization with mixed precision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 17

Shigeng Wang, Chao Li, Yangyuxuan Kang, Jiawei Fan, and Anbang Yao. Attend to your own thoughts: Breaking the barrier for post-training quantization of reasoning llms through the lens of 1.58-bit quantization. *arXiv preprint arXiv:2608.01078*, 2026. 3

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. 3

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025b. 3

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026. 1

Nan Zhang, Eugene Kwek, Yusen Zhang, Ngoc-Hieu Nguyen, Prasenjit Mitra, and Rui Zhang. When reasoning meets compression: Understanding the effects of LLMs compression on large reasoning models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2za3iNkwXn. 3

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025. 7

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2026, 2026. 7

A Appendix

A.1 Calibration Sets

Every stage of the method that touches data uses the same calibration corpus: the second moments of the codebook bank (Stage 1), the Fisher costs of the allocation (Stage 2), and the Hessians and held-out damping selection of the GPTQ refinement (Stage 3). The baselines we compare against are given the identical corpus, so a difference between methods is never a difference in calibration data.

Stratified candidate pool. We first sample a candidate pool eight times larger than the final set, stratified by source so that the pool cannot drift toward whichever domain happens to be most numerous upstream. The pool holds $1{,}024$ documents:

Source	Pool	Selected
`openr1_math_verified`	256	32
`nemotron_ptd_math`	128	16
`nemotron_math_v3_verified`	128	16
`nemotron_ptd_code`	128	16
`nemotron_ptd_stem`	128	16
`nemotron_ptd_chat`	64	8
`nemotron_science`	64	8
`nemotron_chat_reasoning_off`	64	8
`cnn_dailymail_anchor`	64	8
Total	1,024	128

Coverage-greedy selection. Which 128 of the $1{,}024$ documents are kept is decided by routed-expert coverage rather than at random. We run a BF16 forward pass over the whole pool, recording for every document the number of times each (layer, expert) unit is selected by the router, and then fill the per-source quotas greedily under a concave utility. With $c_d \in \mathbb{R}^{L\times E}$ the routing counts of document d and t the counts accumulated by the documents already chosen, the next document from a source with quota remaining is

$$d^\star \;=\; \arg\max_{d}\; \big\langle c_d,\; (t+1)^{-1/2} \big\rangle.$$

The $(t{+}1)^{-1/2}$ weighting makes an expert worth less once it is already well covered, so the selection spends its budget on rarely routed experts instead of re-sampling the popular ones. This matters because the cost sums over the tokens routed to an expert: an expert that no calibration token reaches has no cost estimate at all, and the knapsack would assign it a level on no evidence.

Packing and the loss mask. Selected documents are concatenated into rows of exactly $2{,}048$ tokens; the tail that does not fill a row is dropped. The Fisher gradients of Stage 2 are taken with respect to the mean loss over *assistant* tokens only (the reasoning and answer tokens of the assistant turn plus its closing turn token), computed under a full-context forward pass. Tokens preceding the first assistant turn have no determinable role and are excluded. The Fisher cost is estimated from the first 16 rows, which additionally carry the per-token gradient at each MoE block output, except where noted below.

On-policy construction. The assistant turns of the source datasets were written by other models. Calibrating on that text measures the quantizer against states the target model does not itself visit, and it describes the target's router poorly. The prompts of the same $1{,}024$ documents are re-rendered with the target's chat template and re-answered by the target model in BF16 under its own evaluation sampling protocol; thinking is enabled exactly where the source document had a thinking block, so a reasoning-off document stays reasoning-off. Document $=$ prompt $+$ response, truncated at $2{,}048$ tokens, with the loss mask rebuilt over the new assistant span. Only then does the routing scan run, so both the text and the router that scores coverage belong to the model being quantized.

Held-out split. A disjoint set of 128 documents, sampled from the same sources under the same quotas and capped at 4,096 tokens, is held out. It is used for the candidate comparison that picks the allocation (Stage 2), for the damping grid of Stage 3, and for convergence and rollback decisions; it is never used for computing the costs themselves, and never for evaluation. Benchmark numbers come from frozen checkpoints only.

Sets used per model.

Model	On-policy	Selected documents	Packed rows	Fisher rows
Qwen3.6-35B-A3B	–	128	115	16
Qwen3-Next-80B-A3B	✓	128	120	16
Nemotron-3-Nano-30B-A3B	✓	128	120	120

Every set is built by the same procedure from the same pool and the same quotas; only the model that writes the responses and scores the routing coverage differs. For Nemotron-3-Nano-30B-A3B the Fisher cost is estimated from all 120 rows: with 16 rows its expert ranking had a split-half rank reliability of only 0.47, which rose to 0.69 with 120 rows.

B Supplementary Preliminaries

B.1 Mixed-precision Quantization

The objective in Equation (2) separates over rows,

$$\mathcal{E}(\widehat{W}) = \sum_i (w_i - \widehat{w}_i)^\top S (w_i - \widehat{w}_i), \tag{14}$$

where w_i and $\widehat{w}_i$ denote the i-th rows of W and $\widehat{W}$. It is quadratic in $\widehat{W}$ but combinatorial in the discrete codes: exact minimization over all K^{np} code assignments is intractable. Practical methods therefore decompose the problem into codebook construction, code assignment, and affine-parameter optimization, or optimize the codes greedily (Section 3).

B.2 Exact One-dimensional K-means Clustering

Learning K quantization levels for a set of scalar values $\{x_b\}$ with associated masses $\{m_b\}$ can be formulated as the weighted K-means problem

$$\min_{c_1,\dots,c_K \in \mathbb{R}} F(c) = \sum_b m_b \min_{1 \le k \le K} (x_b - c_k)^2. \tag{15}$$

In general dimensions, K-means is NP-hard (Aloise et al., 2009; Mahajan et al., 2012), and Lloyd's algorithm (Lloyd, 1982), which alternates nearest-centroid assignment with centroid updates, converges only to a local optimum that depends on its initialization. In one dimension, however, the problem admits an exact polynomial-time solution (Fisher, 1958; Bellman, 1973; Wang & Song, 2011; Grønlund et al., 2017), due to the following classical property.

C Quantization-Bank Details

We pool all np normalized values of a matrix into a histogram of $B = 2048$ bins over their $[10^{-4}, 1 - 10^{-4}]$ quantile range (values outside are clamped), so that a few outliers cannot stretch the grid.

Gauge fixing. Equation (1) has an affine redundancy: for any $a \neq 0$ and b, the substitution $c_k \mapsto ac_k + b$, $\alpha_i \mapsto \alpha_i/a$, $\mu_i \mapsto \mu_i - b\alpha_i/a$ leaves every reconstructed weight unchanged. We remove this redundancy by standardizing the codebook according to its priors:

$$\bar{c} = \sum_{k=1}^{K} \pi_k c_k, \qquad c_k \leftarrow \frac{c_k - \bar{c}}{\sqrt{\sum_{r=1}^{K} \pi_r (c_r - \bar{c})^2}}. \tag{16}$$

This keeps the row scales comparable to the original row standard deviations and gives every K a consistent storage convention.

The ternary case ($K = 3$). For three levels, we retain the symmetric alphabet $c = (-1, 0, +1)$. This representation is attractive because symmetric ternary codes can pack five weights into one byte, admit specialized kernels, and the per-row affine parameters already capture much of the flexibility of a learned three-level codebook.

The ternary codes are initialized using the thresholding rule from ternary weight networks (Li et al., 2016) and alternated with the row-wise affine parameters for 15 iterations:

$$T_{ij} = \operatorname{sign}(W_{ij} - \mu_i)\, \mathbb{I}\left[|W_{ij} - \mu_i| > \Delta_i\right], \tag{17}$$

$$\Delta_i = \frac{0.75}{p} \sum_{j=1}^{p} |W_{ij} - \mu_i|, \tag{18}$$

$$\alpha_i = \frac{\sum_j T_{ij}(W_{ij} - \mu_i)}{\sum_j |T_{ij}|}, \tag{19}$$

$$\mu_i \leftarrow \mu_i + \frac{1}{p} \sum_{j=1}^{p} \left(W_{ij} - \mu_i - \alpha_i T_{ij}\right). \tag{20}$$

The scale in Equation (19) is the least-squares fit over the non-zero support, while the offset update absorbs the residual mean. The ternary variant then proceeds directly to Step 3 with $\ell_{ij} = T_{ij}$.

C.1 Step 2: Code Assignment by Coordinate Descent

For $K > 3$, after fixing the codebook, we jointly refine the discrete codes and the row-wise affine parameters by alternating two closed-form updates for 5 iterations, in the spirit of Lloyd's algorithm.

For fixed (μ_i, α_i), each weight is assigned to its nearest reconstruction level:

$$q_{ij} = \arg \min_{1 \le k \le K} \left|W_{ij} - \mu_i - \alpha_i c_k\right|. \tag{21}$$

Let

$$\ell_{ij} = c_{q_{ij}}, \qquad \bar{\ell}_i = \frac{1}{p} \sum_j \ell_{ij}, \qquad \bar{W}_i = \frac{1}{p} \sum_j W_{ij}. \tag{22}$$

For fixed codes, the least-squares affine parameters are

$$\alpha_i = \frac{\sum_j (\ell_{ij} - \bar{\ell}_i)(W_{ij} - \bar{W}_i)}{\sum_j (\ell_{ij} - \bar{\ell}_i)^2}, \tag{23}$$

$$\mu_i = \bar{W}_i - \alpha_i \bar{\ell}_i. \tag{24}$$

Each alternating step exactly minimizes the unweighted row reconstruction error

$$\sum_j \left(W_{ij} - \mu_i - \alpha_i \ell_{ij}\right)^2 \tag{25}$$

with respect to one set of variables while holding the other fixed. The objective is therefore non-increasing, and the iteration converges to a fixed point.

The assignment in Equation (21) is deliberately *unweighted*. Using the full activation-aware objective would couple all coordinates of a row through the off-diagonal entries of S, making independent nearest-level assignment invalid. Instead, activation information enters through the weighted histogram and the affine relocation. The fully coupled code assignment is deferred to Section 3.3, where it is handled with the OBS/GPTQ machinery after the precision of every expert has been selected.

C.2 Step 3: Activation-Aware Relocation of the Row Affine Pair

We freeze the discrete codes and re-optimize (μ_i, α_i) against the *full* second-moment matrix S.

For row i, define

$$J_i(\mu_i, \alpha_i) = (w_i - \mu_i \mathbf{1} - \alpha_i \ell_i)^\top S\,(w_i - \mu_i \mathbf{1} - \alpha_i \ell_i)\,. \tag{26}$$

This is a convex quadratic in the two unknowns α_i and μ_i. Setting both partial derivatives to zero gives the normal equations

$$\begin{bmatrix} \ell_i^\top S \ell_i & \ell_i^\top S \mathbf{1} \\ \mathbf{1}^\top S \ell_i & \mathbf{1}^\top S \mathbf{1} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \mu_i \end{bmatrix} = \begin{bmatrix} \ell_i^\top S w_i \\ \mathbf{1}^\top S w_i \end{bmatrix}. \tag{27}$$

The resulting 2×2 systems are solved for all rows at once. The required quantities can be formed from matrix products involving $S\ell_i$, $S\mathbf{1}$, and Sw_i.

For numerical stability, we regularize the second-moment matrix as

$$S \leftarrow S + \varepsilon I, \qquad \varepsilon = \lambda \,\mathrm{mean}\,(\mathrm{diag}(S))\,, \qquad \lambda = 10^{-4}. \tag{28}$$

If the determinant of the 2×2 Gram matrix in Equation (27) is smaller than $\tau = 10^{-8}$ times the product of its diagonal entries, the system is treated as numerically degenerate and the row retains its Step 2 values.

The resulting affine parameters therefore minimize the expert's activation-weighted output reconstruction error on its routed calibration inputs while accounting for correlations between input coordinates.

D Sensitivity and Allocation Details

D.1 Sensitivity-Aware Mixed-Precision Allocation

Mixed-precision allocation. Not all weights tolerate the same quantization precision. Mixed-precision methods (Dong et al., 2019; 2020; Wang et al., 2019) assign a bitwidth to each block by (i) estimating the increase in task loss $\mathcal{L}$ caused by quantizing that block and (ii) solving a budget-constrained allocation problem.

Second-order sensitivity. Consider a block whose quantization perturbs an intermediate quantity u, either its weights or its output, by δ. A second-order Taylor expansion of the task loss gives

$$\Delta\mathcal{L} \approx g^\top \delta + \frac{1}{2}\delta^\top H \delta, \qquad g = \frac{\partial \mathcal{L}}{\partial u}, \qquad H = \frac{\partial^2 \mathcal{L}}{\partial u^2}. \tag{29}$$

Around a trained model the first-order term is dropped, on the grounds that the gradient vanishes at a stationary point of the training loss (Dong et al., 2019), and the quadratic term serves as the sensitivity. Since the exact Hessian is impractical to compute, mixed-precision methods replace it with tractable proxies such as the Hessian trace (Dong et al., 2020) or the *empirical Fisher*

$$F_{\mathrm{emp}} = \mathbb{E}\left[g g^\top\right], \tag{30}$$

which is constructed from squared gradients and, in practice, further reduced to its diagonal (Martens, 2020; Kunstner et al., 2019).

Applying Equation (29) to $u = (h_t)_t$ and $\delta = (\delta_t)_t$ relies on the following three assumptions.

We assume that the BF16 model is sufficiently close to a stationary point of the training objective such that $\mathbb{E}[g_t] \approx 0$. Accordingly, the first-order Taylor term has approximately zero mean across calibration documents and behaves primarily as finite-sample noise. On our calibration set, the summed absolute magnitude of the first-order term is $2.4\times$ that of the second-order term, while its signed mean is approximately zero. Retaining it would therefore tend to rank quantization levels according to sampling noise. We omit the first-order term. Anchoring every expert and every precision level to the same BF16 reference also places all sensitivity costs on a common scale.

Assumption A5 (Diagonal empirical-Fisher curvature). *Let*

$$g_t^{\Sigma} = \frac{\partial \left(\sum_i \ell_i\right)}{\partial h_t} = T_r g_t \tag{31}$$

denote the gradient of the summed loss. We approximate the curvature of the summed loss at token t by the empirical Fisher $g_t^{\Sigma}(g_t^{\Sigma})^{\top}$ and retain only its diagonal. Because $\mathcal{L}_r$ is an averaged loss, the corresponding curvature is divided by T_r.

Assumption A6 (Additivity). *We approximate the loss increase caused by jointly quantizing multiple experts as the sum of their individual loss increases. This assumption is what reduces mixed-precision assignment to the multiple-choice knapsack problem in Equation (11). It is valid only within the local second-order approximation, i.e., when the individual perturbations remain sufficiently small. The trust region introduced in Section D.2 is designed to enforce this condition.*

Substituting Assumption A5 into the second-order Taylor term gives

$$\frac{1}{2}\sum_t \delta_t^{\top} \frac{\operatorname{diag}\left(g_t^{\Sigma}(g_t^{\Sigma})^{\top}\right)}{T_r}\delta_t = \frac{T_i}{2}\sum_{t\in[T_i]} \langle g_t, \delta_t\rangle^2 . \tag{32}$$

Averaging this quantity over R_F calibration documents yields the cost

$$\Delta(e, K) = \frac{1}{N}\sum_{i=1}^{N} \frac{T_i}{2} \sum_{t\in[T_i]} \langle g_t, \delta_t\rangle^2 . \tag{33}$$

Calibration and Fisher-free alternative. We use $N = 16$ calibration documents, each containing at most 2048 tokens, and apply the language-modeling loss only to assistant tokens. The gradients require one forward–backward pass per calibration document. Afterwards, $\Delta(e, K)$ is evaluated for every K by decoding the corresponding bank entry and executing only the two matrix multiplications of expert e on its routed tokens. Thus, constructing the complete cost table is substantially cheaper than a fine-tuning epoch.

Alongside Equation (33), we also record the plain routed-output reconstruction error

$$\Delta_{\text{out}}(e, K) = \sum_t \|\delta_t\|_2^2 , \tag{34}$$

which provides a Fisher-free alternative for the candidate allocations described in Section D.2.

D.2 Knapsack Allocation

For a Lagrange multiplier $\lambda \geq 0$, the relaxed problem of Equation (11) decouples across experts, and each expert independently selects

$$K_e(\lambda) \in \operatorname*{argmin}_{K\in\{3,\ldots,6\}} \Delta(e, K) + \lambda K. \tag{35}$$

The corresponding total quantization level budget is $B(\lambda) = \sum_{e\in\mathcal{A}} K_e(\lambda)$. As λ increases, larger codebooks are penalized more heavily, and hence $B(\lambda)$ is non-increasing in λ. We therefore use bisection to find the smallest λ such that $B(\lambda) \leq B$. The resulting allocation is optimal for the budget it attains. If a small amount of budget remains unused, we greedily spend it by increasing experts in decreasing order of sensitivity reduction per additional quantization level. Two classical properties of this relaxation underlie the allocation procedure of Section 3.2.

Lemma 2 (Monotonicity; Shoham & Gersho, 1988). *$\bar{b}(\lambda)$ is non-increasing in λ.*

Proposition 1 (Lagrangian optimality; Everett, 1963). *For any $\lambda^{\star} \geq 0$, the allocation $\{K_i(\lambda^{\star})\}_{i=1}^{N}$ minimizes the total cost of Equation (11) among all allocations whose average bitwidth is at most $\bar{b}(\lambda^{\star})$. In particular, it is feasible for Equation (11) whenever $\bar{b}(\lambda^{\star}) \leq \bar{b}$.*

By Lemma D.2, a suitable $\lambda^{\star}$ is located by bisection (Shoham & Gersho, 1988), and Proposition 1 certifies optimality for the discrete budget the resulting allocation actually uses. Because the attainable average bitwidths are discrete, a small amount of budget may remain unused; Section 3.2 describes how this slack is closed.

Trust region. The feasible set $\mathcal{K}_{l,e}$ for an expert is not the entire set $\mathcal{K}$. Equation (33) is a local second-order approximation whose reliability decreases as the quantization perturbation grows. Furthermore, its cost curve as a function of K is not necessarily convex. For approximately 6% of routed experts, the cost at the starting level lies above the chord connecting its two neighboring precision levels. An unrestricted knapsack solver could therefore move such experts toward extreme precisions based on artifacts of the local approximation.

Outside-budget pinning. The pinned precision can be accounted for either *inside* or *outside* the nominal bit budget. When pinning is included inside the budget, the remaining $N - |\mathcal{P}|$ experts are assigned a slightly lower average precision so that the global target $\bar{b}$ is preserved.

Alternatively, the unpinned experts can retain average precision $\bar{b}$. In this case, the resulting average precision over all routed experts is

$$\bar{b}' = \frac{(N - |\mathcal{P}|)\,\bar{b} + |\mathcal{P}|\,b_{K_{\text{pin}}}}{N}. \tag{36}$$

For $\bar{b} = 2$, $N = 10{,}240$, $|\mathcal{P}| = 223$, and $K_{\text{pin}} = 8$, this gives $\bar{b}' = 2.0218$.

E Codebook-Constrained Error Correction Details

We consider a Transformer where each block consists of layer normalization, attention and sparse Mixture-of-Experts (MoE) architecture with shared experts.

Routing-weighted objective. The quantity consumed by the residual stream is the *routing-weighted* expert output $w_{t,e} f_e(x_t)$. For the fused gate/up projection, the squared change in this quantity over the calibration tokens routed to expert e is

$$\sum_t w_{t,e}^2 \left\| \left(W_{gu}^e - \widehat{W}_{gu}^e \right) x_t \right\|_2^2 = \operatorname{tr}\left[\left(W_{gu}^e - \widehat{W}_{gu}^e \right) H_{gu}^e \left(W_{gu}^e - \widehat{W}_{gu}^e \right)^\top \right], \tag{37}$$

Thus, the curvature matrix for the row-wise reconstruction objective is the routing-weight-squared second moment of the expert inputs. A token with routing weight 0.5 contributes one quarter as much as a token with routing weight 1, reflecting the amount of the expert output that reaches the residual stream.

Fixed-codebook objective. Consider one row $w \in \mathbb{R}^p$ with fixed affine parameters (μ, α) and an allocated K-level codebook. The code-assignment problem is

$$\min_{q \in \{0,\dots,K-1\}^p} (w - \widehat{w}(q))^\top H\, (w - \widehat{w}(q)), \qquad \widehat{w}_j(q) = \mu + \alpha c_{q_j}. \tag{38}$$

Step 2 of the quantization bank solved the corresponding problem with $H = I$, in which case the objective decouples into independent nearest-level assignments. With a full curvature matrix H, however, the columns interact: rounding one coordinate upward can be partly compensated by rounding a correlated coordinate downward.

Proposition 3 (Format preservation). *The output of the GPTQ sweep decodes using the same codebook $\{c_k\}_{k=0}^{K-1}$, the same bitwidth, and the same storage layout as the bank variant selected during allocation.*

Proof. Every stored weight is generated by Equation (12) for some $q_j \in \{1, \dots, K\}$ and therefore lies on the reconstruction grid $\{\mu + \alpha c_k\}_{k=1}^K$. The GPTQ compensation step modifies only columns that have not yet been quantized and stored. Hence, neither the code alphabet nor its storage representation is changed. □

Three properties are useful in our setting. First, because Equation (12) is applied to the *updated* column values, a weight may legitimately move to a neighboring code level in response to errors introduced by previously quantized columns. This is precisely the coupled assignment that Step 2 cannot perform; in our experiments, approximately 13% of the codes change during this sweep.

Second, when $H = I$, the off-diagonal compensation vanishes, and the procedure reduces to the independent nearest-level assignment of Equation (21). Step 2 can therefore be viewed as the special case that ignores activation correlations.

Third, after the sweep we explicitly verify grid membership numerically. Consequently, the resulting codes can be consumed directly by the same packed inference kernels as the original bank representation.

E.1 Choosing the damping strength η

The damping parameter η trades fidelity to the calibration curvature against robustness to unseen inputs. Small values closely follow the empirical curvature matrix, whereas large values suppress its anisotropy and move the procedure toward the uncoupled nearest-level assignment in Equation (21).

We select η using calibration data held out from the curvature fit. During selection, the final 15 sequences of the Stage-3 calibration stream are excluded when constructing H. We evaluate $\eta \in \{0.01, 0.03, 0.1, 0.3, 1\}$ using the relative expert-output reconstruction error on these held-out sequences.

To keep the search inexpensive, damping selection is performed using the 64 most frequently routed experts from layers 0, 13, 26, and 39. The selected value of η is then used to rebuild all experts using the full Stage-3 calibration set.

The Stage-3 calibration stream is constructed so that its five source domains remain approximately token-balanced throughout the stream. Its held-out tail therefore remains balanced, preventing the damping selection from disproportionately favoring any single domain.

Affine refit.

$$\begin{bmatrix} \ell^\top H \ell & \ell^\top H \mathbf{1} \\ \mathbf{1}^\top H \ell & \mathbf{1}^\top H \mathbf{1} \end{bmatrix} \begin{bmatrix} \alpha \\ \mu \end{bmatrix} = \begin{bmatrix} \ell^\top H w \\ \mathbf{1}^\top H w \end{bmatrix}. \tag{39}$$

The resulting μ and α are rounded to BF16, matching the storage format of the quantization bank. Thus, the deployed weights remain exactly representable by the packed codebook representation and Proposition 3 continues to hold. Rows with a numerically degenerate system or a non-positive fitted scale retain their affine parameters from the bank.

Scalar gain. In weight space, the least-squares fits in Equations 23, 27, and 39 make the reconstruction residual orthogonal to the fitted representation, leaving little room for an additional scalar correction at the weight level. At $K = 4$, we observe that the corrected expert outputs have, on average, approximately 84% of the norm of their BF16 counterparts. The router nevertheless combines these experts using routing weights computed under the BF16 model. The objective is a scalar quadratic,

$$\gamma^2 \sum_t \|q_t\|_2^2 - 2\gamma \sum_t \langle q_t, r_t \rangle + \text{const}, \tag{40}$$

whose minimizer is exactly Equation (13). Experts with fewer than 64 routed calibration tokens retain $\gamma_e = 1$. Experts receiving no calibration tokens retain their original bank variant. The scalar gain primarily corrects output magnitude rather than direction; by itself it provides limited improvement, but becomes effective after the GPTQ reassignment has removed much of the directional reconstruction error.

E.2 Computational Cost

For each expert, this stage constructs two routing-weighted curvature matrices of size 2048×2048 and 512×512, respectively, and factorizes each matrix once. It then applies the GPTQ sweep to the 1024 rows of the fused gate/up projection and the 2048 rows of the down projection, solves one 2×2 affine system per row, and computes one scalar output gain.

The resulting correction requires approximately 0.5–0.6 seconds per expert for the CPU-side optimization. No GPU is required for the per-expert code-reassignment and affine-fit steps. For the full Qwen 3.6 35B A3B model with 10,240 routed experts, including damping selection and the required calibration passes, the complete stage finishes in approximately half an hour using two GPUs.

Algorithm 1 Stage 1: Quantization-Bank Construction

Require: BF16 routed-expert matrices and routed calibration inputs
Ensure: Bank $\mathcal{B} = \{q_K, c_K, \mu_K, \alpha_K\}$ for all candidate K
for each expert matrix $W \in \{W^e_{gu}, W^e_d\}$ **do**
  Compute the routed-input second moment $S \leftarrow \sum_t x_t x_t^\top$
  Row-normalize W using Equation (4)
  Construct the activation-weighted histogram using $m_{ij} = (\alpha_i^{(0)})^2 S_{jj}$
  Run one exact 1-D dynamic program up to $K_{\max}$
  Recover and gauge-fix the codebook c_K for every candidate K
  for each candidate K **do**
    if $K = 3$ **then**
      Obtain ternary codes using Equations 17–20
    else
      Refine (q, μ, α) by Step-2 coordinate descent
    end if
    Refit (μ, α) using the full S via Equation (27)
    Store $\{q_K, c_K, \mu_K, \alpha_K\}$ in $\mathcal{B}$
  end for
end for
return $\mathcal{B}$

F Entire Quantization Algorithm

Our pipeline consists of three stages: (i) quantization-bank construction, (ii) termination-aware mixed-precision allocation, and (iii) codebook-constrained error correction. Only the routed-expert weights are quantized; all remaining model components stay in BF16. Algorithms 1–3 summarize the full procedure.

Algorithm 2 Stage 2: Termination-Aware Mixed-Precision Allocation

Require: Quantization bank $\mathcal{B}$, calibration documents, routing traces, target bitwidth $\bar{b}$, pin level K_{pin}
Ensure: Allocated level $K_{l,e}$ for every routed expert
Collect BF16 block-output gradients $g_t = \partial \mathcal{L}_r / \partial h_t$
for each layer l, expert e, and candidate level K **do**
  Decode the corresponding bank variant
  Compute $\delta_t = w_{t,e}(f_e^K(x_t) - f_e(x_t))$
  Compute Fisher sensitivity $C[l, e, K]$ using Equation (33)
  Compute the optional Fisher-free output cost $C_{\text{out}}[l, e, K]$
end for
Identify the termination-expert set $\mathcal{P}$ using Equation (8)
Pin $K_{l,e} \leftarrow K_{\text{pin}}$ for all $(l, e) \in \mathcal{P}$
Restrict all remaining experts to the trust region $\mathcal{K}_{l,e}$
Solve the multiple-choice knapsack by Lagrangian bisection
Fill any remaining discrete budget greedily by the largest cost reduction per added bit
Use the Fisher-cost allocation; the held-out calibration NLL is used only to choose the trust region $\mathcal{K}_{l,e}$
return $\{K_{l,e}\}$

F.1 Guarantees

The pipeline combines exact sub-problems with local approximations and greedy procedures. Table 2 summarizes which statements are guaranteed by the corresponding optimization problem and which depend on approximations.

Algorithm 3 Stage 3: Codebook-Constrained Error Correction

Require: Bank $\mathcal{B}$ and allocated level $K_{l,e}$ for each expert
Ensure: Final packed mixed-precision routed experts
  for each routed expert e at its allocated level K **do**
    Construct and damp $H^e_{gu} = \sum_t w^2_{t,e} x_t x_t^\top$
    Reassign W^e_{gu} codes within the fixed codebook using GPTQ/OBS feedback
    Refit $(\mu, \alpha)_{gu}$ using Equation (39)
    Compute corrected intermediate activations $\widetilde{h}_t$
    Construct and damp $H^e_d = \sum_t w^2_{t,e} \widetilde{h}_t \widetilde{h}_t^\top$
    Reassign W^e_d codes within the fixed codebook
    Refit $(\mu, \alpha)_d$
    Fit the scalar output gain γ_e using Equation (13)
    Fold γ_e into the down-projection scale
    Verify that all reconstructed weights remain on the allocated codebook grid
  end for
  return final quantized routed experts

Table 2: Exact and approximate components of the proposed pipeline.

STEP	STATEMENT	STATUS
Codebook	For every K, the dynamic program returns the global minimum of the weighted one-dimensional k-means objective defined on the histogram.	Exact (Lemma 1, Proposition 2)
Step-2 codes	Alternating code assignment and affine fitting never increases the unweighted row reconstruction error and terminates at a fixed point.	Monotone descent
Affine relocation / refit	For fixed codes, the S- or H-weighted row reconstruction error is minimized over (μ, α).	Exact convex 2×2 solve
Sensitivity cost	Equation (33) estimates the loss increase using a BF16-anchored second-order expansion and diagonal empirical-Fisher curvature.	Approximate (A4–A6)
Bit allocation	For a fixed λ, the Lagrangian allocation is optimal among allocations whose average bitwidth does not exceed the discrete bitwidth that it actually attains. Remaining budget slack is filled greedily.	Proposition 1 + greedy slack
Code reassignment	At each processed column, OBS gives the minimum-curvature compensation subject to the current rounding decision; the full sweep is greedy over the column order and preserves the allocated codebook format.	Exact per step; greedy globally
Output gain	For fixed corrected expert outputs, γ_e minimizes the summed squared output mismatch with the BF16 expert.	Exact scalar least squares

F.2 Bits per Weight

Non-uniform quantization learns the levels, typically using k-means over the weight values (Han et al., 2016; Kim et al., 2024), thereby allocating more levels to regions with greater probability mass; ternary quantization (Li et al., 2016; Ma et al., 2024) is the case $K = 3$ with $c = (-1, 0, +1)$ and a per-row scale. The codes require $\log_2 K$ bits per weight, and the two affine parameters, each stored in BF16, contribute a further $32/p$. Ignoring the negligible codebook overhead, the effective bitwidth of Equation (1) is therefore

$$b_{\text{eff}} = \log_2 K + \frac{32}{p}. \tag{41}$$

For an expert matrix using a K-level codebook, the discrete code requires a logical $b_K = \log_2 K$ bits per weight. For non-power-of-two K, we report this logical precision when describing the mixed-precision allocation. The physical packing can differ slightly; for example, five ternary weights can be packed into one byte, corresponding to 1.6 physical bits per weight.

Each matrix also stores one BF16 offset and one BF16 scale for every output row. For a matrix with p input columns, this contributes $b_{\text{affine}} = 32/p$ bits per weight: $32/2048 = 0.015625$ for the fused gate/up projection ($p = 2048$) and $32/512 = 0.0625$ for the down projection ($p = 512$). Because an expert contains twice as many weights in W_{gu} as in W_d, the parameter-weighted affine overhead over a complete expert is $b_{\text{affine}}^{\text{expert}} = (2 \cdot 0.015625 + 0.0625)/3 = 0.03125$ bits per routed-expert weight.

The per-matrix codebook contains only K BF16 scalars and is negligible relative to the matrix itself. The expert gain γ_e is folded into the down-projection scale and therefore adds no separate runtime or storage term.

Unless explicitly stated otherwise, the reported *average bitwidth* refers to the logical code precision

$$\bar{b} = \frac{1}{N} \sum_{l,e} \log_2 K_{l,e}, \tag{42}$$

averaged over the N equally sized routed experts. Including the row-affine overhead, a reported 2.000-bit allocation occupies approximately $2.000 + 0.03125 = 2.03125$ bits per routed-expert weight. Similarly, for the outside-budget pinned allocation in Equation (36), whose logical average precision is $\bar{b}' = 2.0218$, the corresponding storage precision is approximately $2.0218 + 0.03125 = 2.05305$ bits per routed-expert weight.