ERROR-GUIDED REFINEMENT AND COMPOSITION FOR LLM QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training quantization has emerged as an active research area for enabling the efficient deployment of large language models (LLMs). A wide range of quantization methods have been proposed, each based on different design intuitions. We observe that one method can have smaller quantization errors on some weights, while another has smaller errors on others. These differences can affect model performance and may reflect how each method prioritizes different weights. Motivated by this observation, we develop an error-guided approach for both improving individual quantizers and combining their strengths. We first show that selecting more accurate weight approximations from different methods can improve model quality, but introduces additional storage to record the selected method for each weight. To address this issue, we propose an error-guided conversion method that constructs non-uniform variants of existing quantizers using their quantization errors. This conversion provides a common codebook-based representation for different methods. We then combine their codebooks into a single codebook with the same number of entries as an individual non-uniform quantizer, avoiding additional per-weight method labels. Experiments on multiple LLMs show that the proposed conversion can improve perplexity over the original uniform quantizers, while codebook composition provides further gains in several evaluated settings.

1 INTRODUCTION

Post-training quantization (Jacob et al., 2018; Gholami et al., 2021) has become a central technique for enabling the efficient deployment of large language models (LLMs). Given a pretrained model, it aims to reduce memory footprint and inference latency by quantizing model parameters without requiring retraining. This research direction is particularly attractive in large-scale and resource-constrained settings such as edge deployment (Liu et al., 2024a) and on-device inference (Xu et al., 2024). In these scenarios, models are often deployed at scale or under tight hardware constraints, where even small efficiency improvements can translate into substantial practical gains.

A wide range of post-training quantization methods (Frantar et al., 2023; Xiao et al., 2023; Lin et al., 2024; Huang et al., 2025) have been proposed and successfully applied to modern LLMs. At an intuitive level, these methods can be viewed as making different trade-offs in how model weights are preserved during quantization, implicitly answering the question of which parts of the model are the most critical. This perspective underlies their key differences: different methods are guided by distinct design intuitions about weight importance. RTN (Yao et al., 2022) directly rounds weights to a uniform quantization grid. GPTQ (Frantar et al., 2023) uses approximate second-order information to compensate for quantization errors, while AWQ (Lin et al., 2024) protects salient weights according to activation statistics. These methods provide distinct weight approximations that serve as the constituent quantizers in our experiments.

If we view each quantization method as an expert that excels at preserving a particular aspect of the model, a natural question arises: rather than relying on a single expert, can we systematically combine the strengths of multiple experts to further improve quantization quality, analogous to mixture-of-experts models (Shazeer et al., 2017)? That is,

> *Can we go beyond selecting a single quantization strategy, and instead combine multiple strategies to achieve better overall performance?*

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.

This question motivates our work.

1.1 Our Contributions

In this paper, we develop an error-guided approach that allows us to go beyond selecting a single quantization strategy by efficiently combining the strengths of two strategies. Our main contributions are as follows:

- *Complementarity across Quantization Methods.* We show that existing quantization methods can approximate different weights more accurately. Selecting the more accurate approximation for each weight can improve model quality, but requires additional storage for method-selection labels. These findings motivate more efficient ways to combine the strengths of existing quantization methods with lower storage overhead.
- *Error-Guided Conversion.* To combine different quantization methods more efficiently, we first convert them into a common non-uniform format. We propose an error-guided conversion method that uses a source quantizer's element-wise quantization errors, together with model sensitivity, to guide the approximation of the original weights. The resulting representation has a storage cost comparable to that of the source quantizer.
- *Efficient Quantizer Composition.* Building on the common format, we develop a composition procedure to combine the strengths of two converted quantizers. The procedure selects entries from both codebooks and refines them into a single quantizer. The final representation has the same storage cost as an individual converted quantizer under the same configuration, without requiring additional per-weight method labels.
- *Experimental Validation.* We evaluate the proposed approach across multiple LLMs, bit widths, and source quantizer combinations. The results show that conversion improves model performance over the source methods, while composition provides further gains in most evaluated settings under comparable storage budgets.

1.2 Other Related Work

Quantization-Aware Training. Quantization-aware training (QAT) (Jacob et al., 2018) incorporates quantization into training so that model parameters can adapt to quantization noise. PACT (Choi et al., 2018) and LSQ (Esser et al., 2020) learn activation clipping thresholds and quantization step sizes, respectively, while DSQ (Gong et al., 2019) uses a differentiable approximation to quantization. Knowledge distillation (Hinton et al., 2015) provides teacher supervision, and LLM-QAT (Liu et al., 2024b) uses generated data for distillation-based quantization. Related approaches include binarized neural networks (Courbariaux & Bengio, 2016) and BitNet (Wang et al., 2023) for 1-bit training, and QLoRA (Dettmers et al., 2023) for training low-rank adapters over a frozen quantized backbone.

Mixed Precision Quantization. Mixed-precision quantization assigns different bit widths to different layers or weight groups. Existing approaches use Hessian information (Dong et al., 2019) or reinforcement learning (Wang et al., 2019) to determine bit-width assignments. For LLMs, SliM-LLM (Huang et al., 2025) uses weight-group salience to guide this allocation, assigning higher precision to more important groups.

2 Preliminaries

This section formalizes post-training quantization for LLMs. For an arbitrary linear layer, such as an attention projection (Q/K/V) or a feed-forward transformation, let $W \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$ denote its weight matrix and input. The goal is to approximate Wx efficiently using low-bit weight representations while preserving model quality. We describe how weights are stored and reconstructed through a quantized representation and its corresponding linear operator, and formulate their selection as a constrained optimization problem.

Quantized Representations. A quantization method stores weights as low-bit codes Z and decoding parameters ϕ, with reconstructed matrix $D_\phi(Z) \in \mathbb{R}^{m\times n}$. For a b-bit scalar code $z \in \{0, \ldots, 2^b - 1\}$, uniform quantization (e.g. (Jacob et al., 2018)) uses $s(z - \zeta)$, where $s > 0$

and ζ are the scale and zero point. Non-uniform quantization (e.g. (Kim et al., 2024)) maps z to a codebook entry c_z, allowing unevenly spaced reconstruction levels. Decoding parameters may be shared within a tensor, channel, or weight group. More general decoders support vector or additive codebooks and sparse high-precision values.

Quantized Operators. The decoded matrix $D_\phi(Z)$ is used to compute an approximation to the layer output. Some methods additionally change the weight coordinates before quantization and compensate for this change when computing the output. Let $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{n\times n}$ denote the corresponding structured, invertible transformations, such as channel scalings or permutations. Their allowed forms are specified by the quantization scheme.

The codes, decoding parameters, and transformations form a quantization configuration $\Theta = (Z, \phi, A, B)$. For this configuration, we define $\widehat{W}_\Theta = AD_\phi(Z)B$ and $\mathcal{Q}_\Theta(x) = \widehat{W}_\Theta x$, where $\widehat{W}_\Theta$ is the reconstructed weight matrix in the original coordinates, and $\mathcal{Q}_\Theta(x)$ is the output of the quantized layer for input x, approximating the original output Wx.

Optimization Problem. A quantization configuration Θ determines both the reconstructed weight matrix $\widehat{W}_\Theta$ and the storage cost of its representation. Let $\mathcal{F}$ denote the set of allowed configurations, which specifies the code bit widths, decoding rules, parameter-sharing structure, and permitted transformations. Given an error measure $\mathcal{E}$ and a storage budget $\mathcal{B}$, post-training quantization can be formulated as

$$\min_{\Theta\in\mathcal{F}} \mathcal{E}(W, \widehat{W}_\Theta) \qquad \text{s.t.} \quad \mathrm{bits}(\Theta) \le \mathcal{B},$$

where $\mathrm{bits}(\Theta)$ is the total number of bits required to store the codes and all auxiliary information. The objective $\mathcal{E}$ measures the approximation error and may depend on the test data.

Existing Methods. This formulation covers different representations and optimization choices. RTN uses uniform rounding, GPTQ (Frantar et al., 2023) applies second-order error compensation, and AWQ (Lin et al., 2024) selects input-channel scales. OmniQuant (Shao et al., 2024) optimizes clipping and equivalent transformations through block reconstruction. SqueezeLLM (Kim et al., 2024) learns sensitivity-weighted non-uniform codebooks, while SpQR (Dettmers et al., 2024) retains sparse high-precision values.

3 Complementarity of Quantization Methods

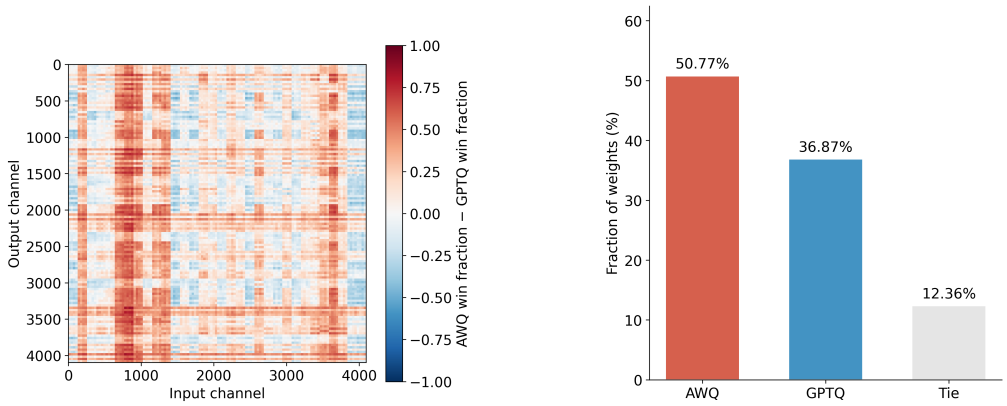
In this section, we examine whether combining the strengths of two quantization methods can improve model quality. For two methods applied to the same weight matrix W, let $\Theta^{(1)}$ and $\Theta^{(2)}$ denote their configurations, with reconstructed weight matrices $\widehat{W}^{(1)} = \widehat{W}_{\Theta^{(1)}}$ and $\widehat{W}^{(2)} = \widehat{W}_{\Theta^{(2)}}$, respectively. We first compare their element-wise quantization errors, then test whether selecting the lower-error approximation at each weight position improves model quality.

3.1 Complementary Weight Approximations

To compare the two methods at individual weight positions, we define their element-wise quantization errors as $\Delta_{ij}^{(\ell)} = \left|W_{ij} - \widehat{W}_{ij}^{(\ell)}\right|$ for each $\ell \in \{1, 2\}$. A smaller $\Delta_{ij}^{(\ell)}$ indicates a more accurate approximation to W_{ij}.

We compare AWQ (Lin et al., 2024) and GPTQ (Frantar et al., 2023) on the 4096×4096 query projection matrix in layer 0 of LLaMA-2-7B. The heatmap in Fig. 1a shows regions favoring each method. As quantified in Fig. 1b, AWQ and GPTQ have smaller errors for 50.77% and 36.87% of the weights, respectively, with equal errors for the remaining 12.36%.

Neither method is uniformly more accurate across this matrix. These complementary approximations motivate selecting the more accurate reconstructed value at each weight position. We next examine whether this selection improves model quality.



(a) Comparison across weight positions.

(b) Fractions of weights favoring each method.

Figure 1: Comparison of element-wise quantization errors between AWQ and GPTQ on the query projection matrix in layer 0 of LLaMA-2-7B. (a) Each cell summarizes 32×32 weights and shows the fraction with smaller AWQ errors minus the fraction with smaller GPTQ errors. Red favors AWQ, blue favors GPTQ, and white indicates equal fractions. The horizontal and vertical axes correspond to input and output channels, respectively. (b) Fractions of individual weights for which AWQ has a smaller error, GPTQ has a smaller error, or both methods have equal errors (Tie).

Table 1: WikiText-2 and C4 perplexity (↓) of AWQ, GPTQ, and their direct composition, MinError. The lowest PPL among the three methods for each model, dataset, and bit width is highlighted in bold.

Model	4-bit						3-bit					
	AWQ		GPTQ		MinError		AWQ		GPTQ		MinError	
	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4	Wiki	C4
LLaMA-2-7B	5.62	8.08	5.62	8.13	**5.56**	**7.99**	6.46	9.23	6.38	9.41	**6.15**	**8.80**
OPT-6.7B	11.05	13.27	11.03	13.38	**10.98**	**13.21**	11.53	13.87	11.83	14.72	**11.34**	**13.63**
Qwen3-8B-Base	7.24	12.80	7.26	12.77	**7.15**	**12.62**	8.86	14.63	8.36	14.30	**8.11**	**13.80**
Mistral-7B-v0.1	5.37	9.18	5.38	9.31	**5.32**	**9.09**	6.03	10.24	6.20	474.86	**5.72**	**9.78**

3.2 Direct Composition

We define *MinError* by selecting the more accurate reconstructed value at each weight position:

$$\widehat{W}_{ij}^{\text{comp}} = \begin{cases} \widehat{W}_{ij}^{(1)}, & \Delta_{ij}^{(1)} \le \Delta_{ij}^{(2)}, \\ \widehat{W}_{ij}^{(2)}, & \Delta_{ij}^{(1)} > \Delta_{ij}^{(2)}. \end{cases}$$

This rule cannot increase the total squared weight reconstruction error relative to either source method. Table 1 shows that MinError[1] improves model quality over both AWQ and GPTQ in the evaluated settings. These gains come with additional storage costs. Decoding requires parameters from both source methods and an explicit binary selection mask, which alone adds one bit per weight. A detailed storage analysis is provided in Appendix B.1. The challenge is therefore how to combine these complementary strengths effectively under a fixed storage budget.

4 Error-Guided Conversion to Non-uniform Quantization

In this section, we propose an error-guided conversion to make existing quantization methods easier to combine. Using a source method's element-wise quantization errors, we approximate the original weights in a common non-uniform format with codebooks and group scales.

[1]See Section 6 for the detailed experimental settings and Appendix A for additional results.

4.1 A COMMON NON-UNIFORM REPRESENTATION

For a given source method, let $\widehat{W}$ denote its reconstructed weight matrix and $\widetilde{W}$ denote the weight matrix after conversion. We represent the converted weights using low-bit indices, non-uniform codebooks, and group scales. An index selects a codebook entry, which is multiplied by the corresponding group scale to reconstruct a weight.

For a weight matrix $W \in \mathbb{R}^{m\times n}$, we partition its mn weight positions into disjoint codebook groups. Let $\mathcal{G} \subseteq \{1,\dots,m\} \times \{1,\dots,n\}$ denote the set of positions in one such group. For b-bit indices, the codebook contains $K = 2^b$ scalar values $\{c_k\}_{k=0}^{K-1}$. The group is further divided into subgroups, with a positive scale t_h for subgroup h. Let $h(i,j)$ denote the subgroup containing weight W_{ij}. For each W_{ij}, the reconstructed value after conversion is $\widetilde{W}_{ij} = t_{h(i,j)}c_{z_{ij}}$ for each $z_{ij} \in \{0,\dots,K-1\}$, where z_{ij} is the b-bit index assigned to the weight. The codebook entries are allowed to be unevenly spaced, while the subgroup scales adjust their magnitudes to fit local weight ranges. Applying this representation to all groups gives the converted weight matrix $\widetilde{W}$. We choose the quantization configuration so that the converted representation has approximately the same total storage cost as the source representation.[2] This common format provides a basis for composition.

4.2 ERROR-GUIDED CONVERSION

The source method's quantization errors indicate which weights it approximates accurately. We use this information to guide the conversion, with the aim of retaining the source method's strengths in the non-uniform representation. For a weight W_{ij}, its absolute quantization error is $\Delta_{ij} = |W_{ij} - \widehat{W}_{ij}|$, where $\widehat{W}_{ij}$ is the reconstructed value from the source method.

To account for the error relative to the weight magnitude, we define $d_{ij} := \eta + (1-\eta)\frac{|W_{ij}|+\epsilon}{|W_{ij}|+\Delta_{ij}+\epsilon}$, where $\eta \in (0,1)$ controls the lower bound of d_{ij} and $\epsilon > 0$ ensures numerical stability. For a fixed weight, a smaller quantization error gives a larger d_{ij}. The coefficient approaches 1 when the error is small relative to the weight magnitude, and approaches η when the relative error is large. This provides a bounded signal of the source method's approximation accuracy.

We further account for model sensitivity using a diagonal empirical Fisher estimate (Kim et al., 2024): $F_{ij} := \frac{1}{N}\sum_{r=1}^{N}\left(\frac{\partial \mathcal{L}_r}{\partial W_{ij}}\right)^2$, where $\mathcal{L}_r$ is the language modeling loss of the original model on the r-th calibration sequence, and N is the number of calibration sequences. The gradients are evaluated at the original model weights. We combine this estimate with the error-based coefficient to obtain $S_{ij} := F_{ij}(1+d_{ij})$. Here, F_{ij} provides a sensitivity estimate, while d_{ij} adjusts its contribution according to the source method's approximation accuracy.

For a codebook group $\mathcal{G}$, we optimize the codeword indices, codebook entries, and positive group scales by minimizing

$$\min_{\{z_{ij}\},\{c_k\},\{t_h\}} \sum_{(i,j)\in\mathcal{G}} S_{ij}\left(W_{ij} - t_{h(i,j)}c_{z_{ij}}\right)^2. \tag{1}$$

The reconstruction target is the original weight W_{ij}, and the source method guides the conversion through the coefficients S_{ij}. Note that these coefficients remain fixed during optimization and are not stored in the final quantized representation.

We optimize Eq. (1) by alternating between code assignments, codebook entries, and group scales. The assignment and codebook updates follow the classical Lloyd procedure (Lloyd, 1982), with sensitivity-weighted clustering used in SqueezeLLM (Kim et al., 2024). Learning codebooks together with scales has also been explored in AQLM (Egiazarian et al., 2024). We apply these established optimization ideas to the conversion objective defined by the error-guided coefficients S_{ij}. Due to space limitations, we defer the details to Appendix C.

[2] At the same code bit width and with the same number of groups, non-uniform quantization generally requires more storage than uniform quantization because its codebook entries must be stored explicitly. We therefore share codebooks across larger groups of weights to keep the total storage cost comparable.

5 FROM CONVERSION TO COMPOSITION

In this section, we combine two error-guided converted quantizers with the same grouping configuration and code bit width into a single non-uniform quantizer. Each provides $K = 2^b$ entries per codebook group; we select K of these $2K$ candidates to initialize a combined codebook. We then determine the codeword assignments and shared group scales, and jointly refine the entries, assignments, and scales. At the same parameter precision, the result has the same storage cost as one converted quantizer, without additional per-weight method labels.

5.1 SELECTING CODEBOOK ENTRIES

For a codebook group $\mathcal{G}$, let $c_k^{(\ell)}$ and $t_h^{(\ell)}$ denote the codebook entries and group scales of converted quantizer $\ell \in \{1, 2\}$, where $k \in \{0, \dots, K-1\}$. We collect the entries from both codebooks into a set $\mathcal{U}$ of $2K$ candidates. A candidate $a = (\ell, k)$ identifies entry k from quantizer ℓ. Its reconstructed value at position (i, j) is $v_{ij,a} = t_{h(i,j)}^{(\ell)} c_k^{(\ell)}$. The source identity is retained during selection so that an entry is evaluated using its corresponding scales. Two entries with the same numerical value can therefore provide different reconstructed values when their source scales differ.

To compare candidates from both quantizers, we use the shared Fisher coefficients F_{ij} introduced in Section 4.2. The source methods have already influenced the converted codebook entries and scales through error-guided conversion. Using F_{ij} at this stage provides a common criterion for selecting among their candidates. For a nonempty candidate set $\mathcal{A} \subseteq \mathcal{U}$, we define its reconstruction error as $E(\mathcal{A}) = \sum_{(i,j)\in\mathcal{G}} F_{ij} \min_{a\in\mathcal{A}} \left(W_{ij} - v_{ij,a}\right)^2$. Thus, each model weight is evaluated against its closest reconstructed value among the candidates in $\mathcal{A}$. Our goal is to retain K candidates while keeping $E(\mathcal{A})$ small. Starting from $\mathcal{A} = \mathcal{U}$, we repeatedly remove the candidate whose deletion causes the smallest increase in reconstruction error:

$$a^\star = \arg\min_{a\in\mathcal{A}} \left[E\left(\mathcal{A} \setminus \{a\}\right) - E(\mathcal{A})\right], \qquad \mathcal{A} \leftarrow \mathcal{A} \setminus \{a^\star\}.$$

The codebook entries and their source scales remain fixed during this process. After a deletion, each model weight can select its closest remaining candidate. The procedure stops when $|\mathcal{A}| = K$.

We remark that this greedy algorithm minimizes the immediate error increase at each step, but does not guarantee a globally optimal subset. However, it provides an efficient selection procedure that avoids enumerating all $\binom{2K}{K}$ subsets of size K.

The retained K entries initialize the combined codebook. At this stage, the candidates are still evaluated with their source scales. In the following, we construct a single quantizer from these entries by determining a common set of group scales and refining the resulting representation.

5.2 REFINING THE COMBINED QUANTIZER

The selected entries were evaluated using the scales of their source quantizers. To obtain a single quantizer, we use these entries to initialize a combined codebook and introduce a common set of group scales. The selected entries are assigned new indices $0, \dots, K-1$, and the scale of subgroup h is initialized as $t_h \leftarrow \sqrt{t_h^{(1)} t_h^{(2)}}$. The codeword assignments are then recomputed using the combined codebook and the initialized scales.

Starting from this initialization, we jointly refine the codebook entries, codeword assignments, and group scales under the same Fisher-weighted criterion used during selection: $\min_{\{z_{ij}\},\{c_k\},\{t_h\}} \sum_{(i,j)\in\mathcal{G}} F_{ij} \left(W_{ij} - t_{h(i,j)} c_{z_{ij}}\right)^2$. We use the alternating updates described in Appendix C, replacing S_{ij} with F_{ij}. The codebook entries can move away from their selected values during refinement, allowing them to adapt to the common group scales and updated assignments.

Applying this procedure to all codebook groups produces the combined weight matrix, with reconstructed entries $\widetilde{W}_{ij}^{\text{comp}} = t_{h(i,j)} c_{z_{ij}}$. Within a codebook group, the final representation stores K codebook entries, one set of group scales, and one b-bit index per model weight. Under the same grouping configuration and parameter precision, its storage cost is identical to that of an individual converted quantizer. No additional per-weight method labels are required.

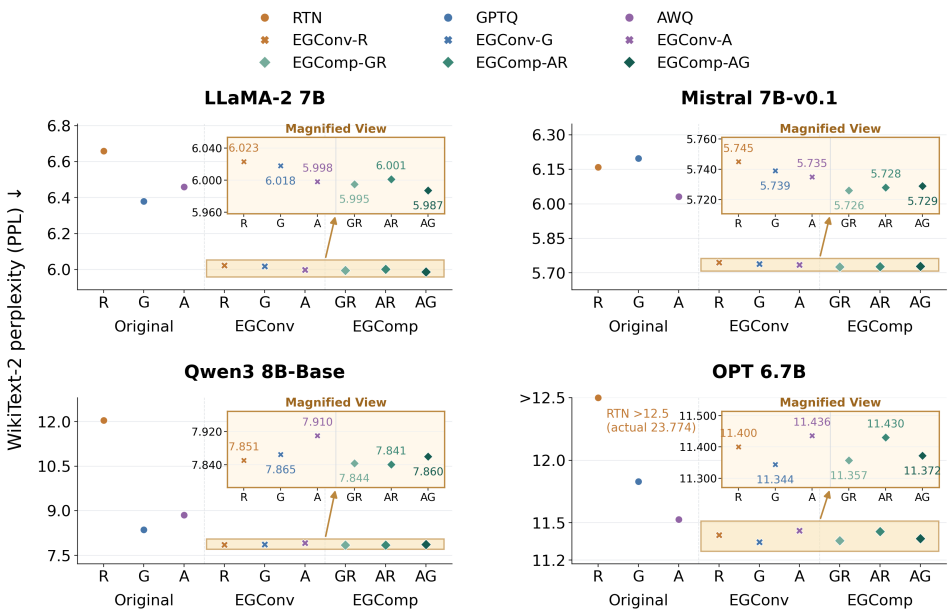


Figure 2: WikiText-2 perplexity (↓) with 3-bit weight indices for AWQ, GPTQ, and RTN, their error-guided conversions (EGConv), and pairwise compositions of the converted quantizers (EGComp).

6 Experiments

This section investigates whether the strengths of individual methods can be preserved during conversion and combined to improve model quality. The experimental setup is as follows:

Models and Evaluation Setup. We evaluate LLaMA-1 (Touvron et al., 2023a), LLaMA-2 (Touvron et al., 2023b), OPT (Zhang et al., 2022), Qwen3 (Team, 2025), and Mistral (Jiang et al., 2023). Perplexity is measured on the WikiText-2 test split and a shard of the C4 English validation split (Merity et al., 2017; Raffel et al., 2020), using non-overlapping sequences of 2048 tokens and discarding the final incomplete sequence. For zero-shot evaluation, we use the LM Evaluation Harness (Gao et al., 2024) on PIQA (Bisk et al., 2020), ARC-Easy and ARC-Challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), and OpenBookQA (Mihaylov et al., 2018). All experiments run on a machine with an NVIDIA RTX Pro 6000 GPU (96 GB memory) and an Intel Core i9-14900K CPU.

Baseline Methods and Hyperparameter Settings. We primarily study the composition of AWQ (Lin et al., 2024) and GPTQ (Frantar et al., 2023), both with a group size of 128. Following prior work, we calibrate them using 128 sequences of 2048 tokens each, randomly sampled from the Pile dataset (Gao et al., 2021) with a fixed seed of 42. EGConv uses codebook groups of 2048 weights and scale subgroups of 128 weights to keep storage comparable to the source quantizers. We set the conversion hyperparameter $\eta = 0.5$ throughout our experiments. We also evaluate RTN (Yao et al., 2022) and its compositions with AWQ and GPTQ to assess generality. We denote the converted AWQ, GPTQ, and RTN quantizers by EGConv-A, EGConv-G, and EGConv-R, respectively, and their pairwise compositions by EGComp-AG, EGComp-AR, and EGComp-GR.

6.1 Gains from Conversion and Composition

AWQ and GPTQ. As shown in Table 2, EGConv-A and EGConv-G reduce perplexity relative to their source methods in all evaluated settings at comparable nominal storage costs. EGComp-AG further improves over the better converted quantizer in 17 of the 20 settings and outperforms both original methods in all 20 settings. The additional gains from composition are generally smaller than those from conversion, but require no extra storage over an individual converted quantizer.

Table 2: WikiText-2 and C4 perplexity (↓) for source quantizers, error-guided conversion, and composition. The best and second-best distinct displayed PPL values are bold and underlined, respectively. Avg. bits denotes nominal packed storage in bits per quantized weight (see Appendix B.2 for detailed storage calculations).

Model	Method	4-bit			3-bit		
		Avg. bits	Wiki	C4	Avg. bits	Wiki	C4
LLaMA-1 7B	AWQ	4.25	5.830	8.273	3.25	6.555	9.336
	GPTQ	4.25	5.841	8.258	3.25	6.586	9.232
	EGConv-A	4.25	<u>5.769</u>	**8.206**	3.19	<u>6.163</u>	<u>8.779</u>
	EGConv-G	4.25	5.782	8.212	3.19	6.176	8.782
	EGComp-AG	4.25	**5.748**	<u>8.208</u>	3.19	**6.162**	**8.776**
LLaMA-2 7B	AWQ	4.25	5.619	8.077	3.25	6.458	9.234
	GPTQ	4.25	5.622	8.126	3.25	6.376	9.406
	EGConv-A	4.25	<u>5.579</u>	8.017	3.19	<u>5.998</u>	8.610
	EGConv-G	4.25	5.581	<u>8.015</u>	3.19	6.018	<u>8.605</u>
	EGComp-AG	4.25	**5.573**	**8.007**	3.19	**5.987**	**8.597**
OPT 6.7B	AWQ	4.25	11.047	13.272	3.25	11.526	13.875
	GPTQ	4.25	11.031	13.377	3.25	11.830	14.717
	EGConv-A	4.25	<u>10.880</u>	13.213	3.19	11.436	13.636
	EGConv-G	4.25	10.908	<u>13.208</u>	3.19	**11.344**	<u>13.634</u>
	EGComp-AG	4.25	**10.865**	**13.203**	3.19	<u>11.372</u>	**13.619**
Qwen3 8B-Base	AWQ	4.25	7.242	12.796	3.25	8.856	14.627
	GPTQ	4.25	7.259	12.769	3.25	8.359	14.297
	EGConv-A	4.25	7.176	12.676	3.19	7.910	13.758
	EGConv-G	4.25	**7.166**	<u>12.674</u>	3.19	<u>7.865</u>	<u>13.720</u>
	EGComp-AG	4.25	<u>7.167</u>	**12.664**	3.19	**7.860**	**13.702**
Mistral 7B-v0.1	AWQ	4.25	5.367	9.178	3.25	6.032	10.240
	GPTQ	4.25	5.376	9.311	3.25	6.197	474.856
	EGConv-A	4.25	5.361	9.143	3.19	<u>5.735</u>	<u>9.797</u>
	EGConv-G	4.25	<u>5.353</u>	<u>9.141</u>	3.19	5.739	9.804
	EGComp-AG	4.25	**5.349**	**9.137**	3.19	**5.729**	**9.789**

Compositions Involving Other Methods. Fig. 2 reports 3-bit results for RTN and its compositions with AWQ and GPTQ. EGConv-R improves over RTN on all four evaluated models. Source rankings can also change after conversion: on Qwen3-8B-Base, RTN has the highest initial perplexity, yet EGConv-R outperforms both EGConv-A and EGConv-G, with EGComp-AR and EGComp-GR providing further gains. Thus, a source method with worse initial performance can still provide a useful basis for conversion and composition.

Remark. We further observe that the additional gains from composition depend on the model, the source pair, and the evaluation dataset. As shown in Fig. 2, all three compositions outperform both of their converted components on Mistral-7B-v0.1 when evaluated on WikiText-2, whereas none improves over its better converted component on OPT-6.7B. On C4, however, all three compositions outperform both of their converted components across all evaluated models (see Appendix D). We present the WikiText-2 results in the main text to illustrate this variation, including settings where composition does not provide further gains.

6.2 Effect of Error Guidance

To examine the contribution of the source method's quantization errors, we compare error-guided conversion with a variant that uses only the Fisher coefficients. We denote this variant as EGConv-None. It replaces $S_{ij} = F_{ij}(1 + d_{ij})$ with F_{ij} in the conversion objective, removing the source-specific error information while retaining the sensitivity estimate.

Table 3: WikiText-2 test perplexity (↓) for the error-guidance ablation. EGConv-None uses only Fisher coefficients, without the delta-derived factor. The best and second-best distinct displayed PPL values are bold and underlined, respectively.

Bits	Method	LLaMA-2 7B	OPT 6.7B	Qwen3 8B-Base	Mistral 7B-v0.1
4-bit	EGConv-None	5.578	10.888	**7.166**	5.356
	EGConv-A	5.579	10.880	7.176	5.361
	EGConv-G	5.581	10.908	**7.166**	5.353
	EGComp-AG	**5.573**	**10.865**	7.167	**5.349**
3-bit	EGConv-None	6.015	11.448	7.905	5.773
	EGConv-A	5.998	11.436	7.910	5.735
	EGConv-G	6.018	**11.344**	7.865	5.739
	EGComp-AG	**5.987**	11.372	**7.860**	**5.729**

Table 4: Zero-shot accuracy (%, ↑) under 3-bit weight quantization. Avg. is the unweighted mean across seven tasks.

Method	PIQA	ARC-E	ARC-C	BoolQ	Hella Swag	Wino Grande	OBQA	Avg.
LLaMA-2 7B								
AWQ	**77.31**	68.98	**43.17**	69.30	73.22	67.72	**43.00**	63.24
GPTQ	76.66	70.96	42.15	72.57	71.49	66.77	39.00	62.80
EGComp-AG	75.79	**71.97**	40.44	**77.13**	**73.41**	**68.59**	41.40	**64.10**
OPT-6.7B								
AWQ	75.30	57.49	31.31	61.80	**64.77**	62.04	**36.00**	55.53
GPTQ	75.41	56.78	32.17	61.96	62.22	61.17	34.60	54.90
EGComp-AG	**75.46**	**65.24**	**33.02**	**65.93**	64.24	**63.54**	34.00	**57.35**
Qwen3-8B-Base								
AWQ	77.58	77.27	**53.07**	77.28	74.56	65.51	38.60	66.27
GPTQ	**78.29**	77.61	51.45	81.50	**74.65**	67.88	40.80	67.45
EGComp-AG	77.69	**78.54**	52.56	**82.39**	74.63	**71.67**	**41.40**	**68.41**
Mistral-7B-v0.1								
AWQ	**80.74**	76.43	50.00	81.25	**78.47**	71.51	44.20	68.94
GPTQ	50.38	25.51	28.67	43.61	26.04	49.57	25.00	35.54
EGComp-AG	79.38	**79.12**	**50.85**	**82.14**	78.32	**74.82**	**45.00**	**69.95**

Table 3 reports the resulting WikiText-2 perplexity at 3-bit and 4-bit precision. In six of the eight evaluated settings, the coefficients d_{ij} from at least one source method improve conversion over EGConv-None. However, the gains vary with the source method, and error guidance leads to higher perplexity in some cases. The quantization errors of a method reflect its own approximation choices, which may not always indicate which weights should receive greater emphasis during conversion.

6.3 Downstream Task Accuracy

To examine whether the improvements extend to downstream task performance, we evaluate EGComp-AG on seven zero-shot tasks and compare it with the original AWQ and GPTQ quantizers. Table 4 reports the task-level accuracy and the average across all seven tasks for four models under 3-bit quantization. EGComp-AG achieves higher average accuracy than both source quantizers on all four evaluated models. The improvements over the better source quantizer range from 0.86 to 1.82 percentage points. However, the gains vary across tasks. For example, on LLaMA-2-7B, EGComp-AG improves accuracy on BoolQ and WinoGrande, but reduces it on PIQA and ARC-Challenge relative to both source methods. Thus, the final quantizer improves average downstream performance in the evaluated settings, while individual tasks may benefit differently.

AI USE STATEMENT

We used generative AI tools to assist with implementing the methods and experimental code, and searching the literature. AI-generated suggestions were used only as auxiliary references and were independently reviewed, tested, and verified by the authors before adoption. Relevant literature and citations were checked against the original sources, and all methodological decisions, experimental results, analyses, and conclusions were determined and validated by the authors. The authors take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

The conversion and composition procedures are described in Sections 4 and 5, with codebook initialization and alternating optimization detailed in Appendix C. The datasets, evaluation protocols, quantization settings, and computational resources are described in Section 6. To support reproduction of the main perplexity comparison in Table 2, we provide anonymized source code as supplementary material. The implementation includes source-quantizer preparation, empirical Fisher estimation, error-guided conversion, composition, and WikiText-2 and C4 perplexity evaluation.

REFERENCES

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *AAAI*, pp. 7432–7439. AAAI Press, 2020.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: parameterized clipping activation for quantized neural networks. *CoRR*, abs/1805.06085, 2018.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *NAACL-HLT (1)*, pp. 2924–2936. Association for Computational Linguistics, 2019.

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. *CoRR*, abs/1803.05457, 2018.

Matthieu Courbariaux and Yoshua Bengio. BinaryNet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *NeurIPS*, 2023.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *ICLR*. OpenReview.net, 2024.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: hessian aware quantization of neural networks with mixed-precision. In *ICCV*, pp. 293–302. IEEE, 2019.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12284–12303. PMLR / OpenReview.net, 2024.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *ICLR*. OpenReview.net, 2020.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: accurate quantization for generative pre-trained transformers. In *ICLR*. OpenReview.net, 2023.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *CoRR*, abs/2103.13630, 2021.

Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *ICCV*, pp. 4851–4860. IEEE, 2019.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qinshuo Liu, Xianglong Liu, Luca Benini, Michele Magno, Shiming Zhang, and Xiaojuan Qi. SliM-LLM: Salience-driven mixed-precision quantization for large language models. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

Yerlan Idelbayev, Pavlo Molchanov, Maying Shen, Hongxu Yin, Miguel Á. Carreira-Perpiñán, and José M. Álvarez. Optimal quantization using scaled codebook. In *CVPR*, pp. 12095–12104. Computer Vision Foundation / IEEE, 2021.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, pp. 2704–2713. Computer Vision Foundation / IEEE Computer Society, 2018.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *CoRR*, abs/2310.06825, 2023.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 23901–23923. PMLR / OpenReview.net, 2024.

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. In *MLSys*. mlsys.org, 2024.

Hou-I Liu, Marco Galindo, Hongxia Xie, Lai-Kuan Wong, Hong-Han Shuai, Yung-Hui Li, and Wen-Huang Cheng. Lightweight deep learning for resource-constrained environments: A survey. *ACM Comput. Surv.*, 56(10):267, 2024a.

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 *ACL (Findings)*, volume ACL 2024 of *Findings of ACL*, pp. 467–484. Association for Computational Linguistics, 2024b.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR (Poster)*. OpenReview.net, 2017.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *EMNLP*, pp. 2381–2391. Association for Computational Linguistics, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

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

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *ICLR*. OpenReview.net, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR (Poster)*. OpenReview.net, 2017.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b.

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. *CoRR*, abs/2310.11453, 2023.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: hardware-aware automated quantization with mixed precision. In *CVPR*, pp. 8612–8620. Computer Vision Foundation / IEEE, 2019.

Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023.

Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. On-device language models: A comprehensive review. *CoRR*, abs/2409.00088, 2024.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers. In *NeurIPS*, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *ACL (1)*, pp. 4791–4800. Association for Computational Linguistics, 2019.

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

A Additional Results on Direct Composition

This section presents additional results for the direct-composition method introduced in Section 3.2. R, G, and A denote RTN, GPTQ, and AWQ, respectively. MinError-GR, MinError-AR, and MinError-AG select the lower-error weight approximation from GPTQ+RTN, AWQ+RTN, and AWQ+GPTQ, respectively. MinError-AGR selects from all three quantizers. The bit-width and group-size settings refer to the source quantizers; MinError compositions additionally require method-selection metadata.

A.1 Language Modeling Results

Tables 5 and 6 report WikiText-2 and C4 perplexity across eight models at 3-bit and 4-bit precision. The lowest quantized PPL for each model and precision setting is highlighted in bold.

Table 5: Perplexity (↓) on WikiText-2 across different model families. All source quantizers use a group size of 128. Precision labels exclude the additional method-selection metadata used by MinError compositions. The best quantized result under each precision setting is highlighted in bold.

Bits	Method	LLaMA 1-7B	LLaMA 2-7B	LLaMA 1-13B	LLaMA 2-13B	OPT 6.7B	OPT 13B	Qwen3 8B Base	Mistral 7B v0.1
FP16	–	5.68	5.47	5.09	4.88	10.86	10.13	6.99	5.25
INT4	RTN	5.96	5.73	5.26	4.98	11.17	10.31	8.09	5.43
	GPTQ	5.84	5.62	5.19	4.99	11.03	10.23	7.26	5.38
	AWQ	5.83	5.62	5.18	4.98	11.05	10.27	7.24	5.37
	MinError-GR	5.89	5.66	5.19	4.95	11.00	10.28	7.40	5.37
	MinError-AR	5.80	5.59	5.18	4.96	10.99	10.21	7.17	5.34
	MinError-AG	**5.77**	5.56	**5.14**	4.94	10.98	**10.20**	7.15	5.32
	MinError-AGR	**5.77**	**5.55**	**5.14**	**4.93**	**10.97**	**10.20**	**7.11**	**5.31**
INT3	RTN	7.01	6.66	5.89	5.52	23.77	44.51	12.05	6.16
	GPTQ	6.59	6.38	5.66	5.43	11.83	10.55	8.36	6.20
	AWQ	6.55	6.46	5.65	5.44	11.53	10.62	8.86	6.03
	MinError-GR	6.58	6.31	5.61	5.32	12.25	11.36	9.49	5.83
	MinError-AR	6.44	6.30	5.57	5.36	11.42	10.47	8.42	5.86
	MinError-AG	6.31	6.15	5.45	5.25	11.34	10.40	8.11	5.72
	MinError-AGR	**6.27**	**6.10**	**5.43**	**5.22**	**11.28**	**10.38**	**8.04**	**5.68**

Table 6: Perplexity (↓) on C4 across different model families. All source quantizers use a group size of 128. Precision labels exclude the additional method-selection metadata used by MinError compositions. The best quantized result under each precision setting is highlighted in bold.

Bits	Method	LLaMA 1-7B	LLaMA 2-7B	LLaMA 1-13B	LLaMA 2-13B	OPT 6.7B	OPT 13B	Qwen3 8B Base	Mistral 7B v0.1
FP16	–	8.07	7.87	7.41	7.17	13.13	12.51	12.40	8.99
INT4	RTN	8.43	8.19	7.56	7.33	13.72	12.86	14.76	9.45
	GPTQ	8.26	8.13	7.55	7.30	13.38	12.59	12.77	9.31
	AWQ	8.27	8.08	7.53	7.28	13.27	12.62	12.80	9.18
	MinError-GR	8.30	8.07	7.51	7.27	13.50	12.71	13.02	9.38
	MinError-AR	8.23	8.03	7.51	7.26	13.23	12.59	12.67	9.13
	MinError-AG	8.19	**7.99**	7.49	7.24	13.21	12.57	12.62	9.09
	MinError-AGR	**8.18**	8.03	**7.48**	**7.23**	**13.20**	**12.56**	**12.59**	**9.08**
INT3	RTN	9.87	9.57	8.48	8.06	26.78	44.53	20.78	10.68
	GPTQ	9.23	9.41	8.18	7.98	14.72	12.98	14.30	474.86
	AWQ	9.34	9.23	8.14	7.88	13.87	13.07	14.63	10.24
	MinError-GR	9.28	9.02	8.10	7.79	14.88	14.06	16.27	10.11
	MinError-AR	9.16	9.04	8.02	7.78	13.72	12.95	14.15	9.96
	MinError-AG	**8.96**	**8.80**	7.89	7.65	13.63	12.88	13.80	9.78
	MinError-AGR	9.16	9.04	**7.85**	**7.61**	**13.59**	**12.84**	**13.66**	**9.68**

A.2 Zero-Shot Task Results

Table 7 reports seven task-level scores and their averages for LLaMA-2-7B, OPT-6.7B, Qwen3-8B-Base, and Mistral-7B-v0.1. Quantized methods use 3-bit source weight codes and a group size of 128. The highest quantized average for each model is highlighted in bold.

Table 7: Zero-shot accuracy (%) of LLaMA-2 7B, OPT-6.7B, Qwen3-8B-Base, and Mistral-7B-v0.1 under 3-bit weight-only quantization. The best quantized average for each model is highlighted in bold.

Model	Method	PIQA	ARC-E	ARC-C	BoolQ	Hella Swag	Wino Grande	OBQA	Avg
LLaMA-2 7B	FP16	78.84	74.87	45.73	79.08	76.23	69.14	43.80	66.81
	RTN	77.15	70.96	42.24	72.23	74.01	68.82	42.60	64.00
	GPTQ	76.66	70.96	42.15	72.57	71.49	66.77	39.00	62.80
	AWQ	77.31	68.98	43.17	69.30	73.22	67.72	43.00	63.24
	MinError-GR	77.58	71.21	43.77	72.17	74.68	68.03	42.00	64.21
	MinError-AR	78.13	71.09	44.97	71.04	73.94	68.19	43.40	64.39
	MinError-AG	78.13	71.09	43.43	72.60	74.26	67.64	41.80	64.13
	MinError-AGR	77.97	71.93	44.03	73.21	74.51	67.48	42.00	**64.45**
OPT 6.7B	FP16	76.71	60.44	34.22	67.09	67.15	65.59	37.40	58.37
	RTN	72.47	52.53	30.89	59.39	57.83	59.19	34.40	52.38
	GPTQ	75.41	56.78	32.17	61.96	62.22	61.17	34.60	54.90
	AWQ	75.30	57.49	31.31	61.80	64.77	62.04	36.00	55.53
	MinError-GR	75.68	56.23	31.48	61.93	64.19	63.22	33.80	55.22
	MinError-AR	76.39	58.50	32.25	60.70	65.62	63.93	36.80	56.31
	MinError-AG	76.33	57.91	32.51	64.13	66.08	63.30	37.00	**56.75**
	MinError-AGR	76.33	58.25	32.51	63.55	65.94	63.14	37.00	56.67
Qwen3 8B-Base	FP16	79.38	80.05	57.00	83.00	78.64	72.45	42.00	70.36
	RTN	75.30	76.30	50.60	71.65	69.93	64.25	40.00	64.00
	GPTQ	78.29	77.61	51.45	81.50	74.65	67.88	40.80	67.45
	AWQ	77.58	77.27	53.07	77.28	74.56	65.51	38.60	66.27
	MinError-GR	77.80	76.98	51.19	80.80	73.32	68.82	43.00	67.42
	MinError-AR	78.45	79.34	54.44	81.90	75.37	67.88	39.80	68.17
	MinError-AG	78.94	79.46	53.84	81.22	76.04	70.17	41.40	68.72
	MinError-AGR	78.67	79.50	54.35	83.03	76.32	69.53	40.60	**68.86**
Mistral 7B-v0.1	FP16	82.64	79.92	54.01	84.28	81.14	74.98	46.40	71.91
	RTN	80.74	75.38	47.35	80.43	78.49	71.43	44.60	68.35
	GPTQ	50.38	25.51	28.67	43.61	26.04	49.57	25.00	35.54
	AWQ	80.74	76.43	50.00	81.25	78.47	71.51	44.20	68.94
	MinError-GR	81.83	76.09	50.26	81.56	79.53	71.82	45.20	69.47
	MinError-AR	80.58	75.97	50.09	81.56	79.19	72.06	44.20	69.09
	MinError-AG	81.77	77.36	51.71	81.77	79.40	72.38	45.60	70.00
	MinError-AGR	81.88	77.27	51.54	82.63	79.59	73.24	44.80	**70.14**

B STORAGE ACCOUNTING

We analyze explicitly specified theoretical packed representations. Let N denote the number of quantized weights, b the number of bits per weight code or codeword index, and $g = 128$ the scale group size. We assume complete groups and tightly packed codes or indices. All scales and codewords are FP16. For the AWQ and GPTQ representations considered here, each G128 group stores one FP16 scale and one FP16 zero-point. The 16-bit zero-point is a storage assumption, not an intrinsic requirement of either method. All estimates exclude tail-group padding, unquantized parameters, file metadata, alignment, and any additional index mappings or unabsorbed channel scales.

B.1 MINERROR

We consider the two-source MinError composition of AWQ and GPTQ.

MinError stores one selected b-bit code per weight, the scale and zero-point arrays of both sources, and a packed binary source-selection mask with one bit per weight. Thus, it does not retain two complete weight-code arrays. The mask determines which source parameters are used to decode each selected code. We assume that the source reconstructions are expressible using these codes and group parameters in a common weight coordinate system.

Under these assumptions, the storage costs in bits are

$$B_{\text{single}} = \underbrace{bN}_{\text{codes}} + \underbrace{16N/g}_{\text{scales}} + \underbrace{16N/g}_{\text{zero-points}},$$

$$B_{\text{MinError}} = \underbrace{bN}_{\text{selected codes}} + \underbrace{32N/g}_{\text{two scale arrays}} + \underbrace{32N/g}_{\text{two zero-point arrays}} + \underbrace{N}_{\text{source mask}}.$$

Dividing by N gives the per-weight contributions in Table 8.

Table 8: Theoretical storage in bits per quantized weight under the stated G128 format. Scales and zero-points are FP16, and source masks are bit-packed. MinError-AG combines AWQ and GPTQ.

Bits	Method	Codes	Scales	Zeros	Mask	Total
4-bit	AWQ	4	0.125	0.125	0	4.25
	GPTQ	4	0.125	0.125	0	4.25
	MinError-AG	4	0.250	0.250	1	5.50
3-bit	AWQ	3	0.125	0.125	0	3.25
	GPTQ	3	0.125	0.125	0	3.25
	MinError-AG	3	0.250	0.250	1	4.50

Relative to a single quantizer, MinError adds

$$\underbrace{0.125}_{\text{extra scales}} + \underbrace{0.125}_{\text{extra zero-points}} + \underbrace{1}_{\text{mask}} = 1.25 \text{ bits per weight.}$$

Different mask encodings or retention of both complete source code arrays lead to different storage costs.

B.2 EGConv and EGComp

Under the same storage assumptions, we detail the nominal average bit-width reported in Table 2. Let $G = 2048$ be the codebook group size and $K = 2^b$ the number of codewords per codebook. The AWQ/GPTQ baseline occupies $B_{\text{single}}/N = b + 0.25$ bits per weight.

EGConv stores one b-bit index per weight, K FP16 codewords per G2048 group, and one FP16 scale per G128 group. EGComp stores the same final representation after composition: one codebook of K codewords per large group and one scale per small group. Thus, both methods have average storage

$$\bar{b}_{\text{EGConv/EGComp}} = b + \frac{16K}{2048} + \frac{16}{128}.$$

Fisher weights and delta-derived importance coefficients are used during construction and are not required in the final decoding format.

Table 9: Nominal storage breakdown in bits per quantized weight. All scales and codewords are FP16; AWQ/GPTQ zero-points are also assumed to occupy 16 bits. Zeros denote zero-point storage, and centers denote codebook storage. Zero entries indicate that the corresponding component is not stored.

Bits	Method	Codes	Scales	Zeros	Centers	Total
4-bit	AWQ	4	0.125	0.125	0	4.25
	GPTQ	4	0.125	0.125	0	4.25
	EGConv-A	4	0.125	0	0.125	4.25
	EGConv-G	4	0.125	0	0.125	4.25
	EGComp-AG	4	0.125	0	0.125	4.25
3-bit	AWQ	3	0.125	0.125	0	3.25
	GPTQ	3	0.125	0.125	0	3.25
	EGConv-A	3	0.125	0	0.0625	3.1875
	EGConv-G	3	0.125	0	0.0625	3.1875
	EGComp-AG	3	0.125	0	0.0625	3.1875

At 4 bits, each G2048 group in EGConv or EGComp stores 2048 indices, 16 centers, and 16 scales. These occupy $8192 + 256 + 256 = 8704$ bits, or $8704/2048 = 4.25$ bits per weight. For AWQ or GPTQ under the stated format, the same 2048 weights require 8192 code bits, 256 scale bits, and 256 zero-point bits. The totals coincide even though the decoding parameters differ. At 3 bits, the EGConv/EGComp codebook has eight centers, reducing its contribution to $128/2048 = 0.0625$ bits per weight. The total is 3.1875, rounded to 3.19 in the main table. These nominal values do not depend on model size under the complete-group assumption.

C Optimizing Codebooks and Scales

This section describes how to optimize Eq. (1) by alternating between code assignments, codebook entries, and group scales.

Initialization. We first initialize the group scales, then hold them fixed to construct the initial codebook. Let $\mathcal{H}_h = \{(i,j) \in \mathcal{G} : h(i,j) = h\}$ denote the positions in scale subgroup h. The scale is initialized to the root mean square of the model weights in this subgroup:

$$t_h = \sqrt{\frac{1}{|\mathcal{H}_h|} \sum_{(i,j)\in\mathcal{H}_h} W_{ij}^2}.$$

For an all-zero subgroup, we set $t_h = 1$ instead.

With these scales fixed, we normalize the model weights within the codebook group $\mathcal{G}$ and define the corresponding clustering coefficients:

$$u_{ij} = \frac{W_{ij}}{t_{h(i,j)}}, \qquad \omega_{ij} = S_{ij} t_{h(i,j)}^2.$$

Here, u_{ij} is the normalized value of W_{ij}, and ω_{ij} is its corresponding coefficient, which will be used in the following.

We obtain the initial codebook by clustering the normalized values $\{u_{ij}\}$ into K clusters. Using $\{\omega_{ij}\}$ as the clustering coefficients, we formulate this as

$$\min_{\{z_{ij}\},\{c_k\}} \sum_{(i,j)\in\mathcal{G}} \omega_{ij} \left(u_{ij} - c_{z_{ij}}\right)^2.$$

We use weighted K-means (Lloyd, 1982) to approximately solve this problem. The resulting K cluster centers serve as the initial codebook entries $\{c_k\}$.

We remark that the factor $t_{h(i,j)}^2$ accounts for the normalization: a difference of $u_{ij} - c_k$ in the normalized space corresponds to a reconstruction error of $t_{h(i,j)}(u_{ij} - c_k)$ in the original model weights. Thus, with the scales fixed, the clustering objective is equivalent to the conversion objective (Eq. (1)).

Alternating Updates. After initialization, we alternate between the following three updates, holding the other variables fixed at each step.

(1) *Update Code Assignments.* With the codebook and scales fixed, we assign each model weight to its nearest reconstructed value:

$$z_{ij} \leftarrow \arg\min_{k\in\{0,\ldots,K-1\}} \left(W_{ij} - t_{h(i,j)} c_k\right)^2.$$

Since S_{ij} is nonnegative and independent of k, this choice minimizes the corresponding weighted error.

(2) *Update Codebook Entries.* With the assignments and scales fixed, we update the codebook entries by weighted least squares:

$$c_k \leftarrow \left(\sum_{\substack{(i,j)\in\mathcal{G}\\ z_{ij}=k}} S_{ij} t_{h(i,j)} W_{ij}\right) \Big/ \left(\sum_{\substack{(i,j)\in\mathcal{G}\\ z_{ij}=k}} S_{ij} t_{h(i,j)}^2\right).$$

If the denominator is zero, the codebook entry retains its previous value.

(3) *Update Group Scales.* With the assignments and codebook fixed, we update the scales as

$$t_h \leftarrow \max\left\{\tau, \left(\sum_{(i,j)\in\mathcal{H}_h} S_{ij} W_{ij} c_{z_{ij}}\right) \Big/ \left(\sum_{(i,j)\in\mathcal{H}_h} S_{ij} c_{z_{ij}}^2\right)\right\}.$$

where $\tau > 0$ is a small lower bound on the scale. This is a weighted least-squares update with a positive lower bound, corresponding to the weighted form of the fixed-assignment scale update studied by Idelbayev et al. (2021). If the denominator is zero, the scale retains its previous value.

The coefficients S_{ij} remain fixed throughout these updates. Applying the procedure independently to all codebook groups produces the converted weight matrix $\widetilde{W}$. The resulting representation stores only the codeword indices, codebooks, and group scales. This conversion provides a common non-uniform format for any source method that produces reconstructed weights. Our experiments further show that the conversion preserves model quality in the evaluated settings and improves it in most cases.

D Additional C4 Results on Conversion and Composition

Figure 3 complements the WikiText-2 results in Fig. 2 with C4 evaluation of the same 3-bit methods on LLaMA-2-7B, OPT-6.7B, Qwen3-8B-Base, and Mistral-7B-v0.1. Each panel compares the original source quantizers, their EGConv variants, and the three pairwise EGComp variants. The magnified views make the smaller differences between conversion and composition visible alongside the larger improvements over the source quantizers.

On all four models, each EGConv variant reduces C4 perplexity relative to its corresponding source quantizer. Each of EGComp-AG, EGComp-AR, and EGComp-GR further improves over both of its converted components. For example, on OPT-6.7B, EGComp-AG achieves a perplexity of 13.619, compared with 13.636 for EGConv-A and 13.634 for EGConv-G. These results show that composition provides additional gains in the evaluated C4 settings, although the gains over conversion are smaller than those from conversion over the source quantizers.

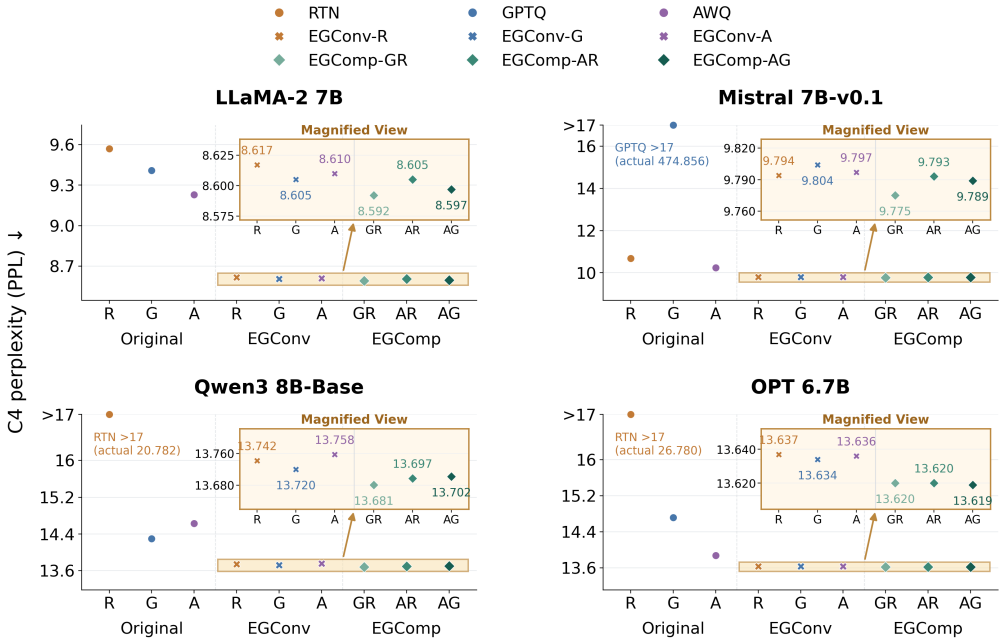


Figure 3: C4 perplexity (↓) with 3-bit weight indices for AWQ, GPTQ, and RTN, their error-guided conversions (EGConv), and pairwise compositions of the converted quantizers (EGComp). Insets magnify the highlighted regions using separate vertical scales. Results above 17 are shown at the upper boundary with their actual values.