QUENCH: PRUNE ITERATIVELY AND QUANTISE ONCE FOR TRAINING-FREE LLM COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models require substantial memory to store their weights. Pruning removes selected weights and quantisation represents them with fewer bits, but both can reduce accuracy. A small low-rank correction can recover some of the lost accuracy without fine-tuning. We introduce QUENCH, a training-free pipeline that designs pruning and quantisation around such a correction. We target the linear layers of the transformer architecture. QUENCH alternates pruning and activation-aware correction, enabling the sparse weights to respond to the previous correction. It then quantises the surviving weights once with elastic-bucket quantisation (EBQ), which learns representative weight values (codebooks) for each output channel, and refits the correction. On LLaMA-3-8B with 2:4 sparsity and 3-bit values, QUENCH reaches WikiText2 perplexity 12.01 and 34.22% ARC-Challenge accuracy, against 12.60 and 31.22% reported for EoRA with 2:4 sparsity and 4-bit weights. We also weigh the improvement from activation weighting against low-rank correction. It improves standalone codebooks by 12.29 points, but provides no consistent additional benefit after a rank-128 EoRA correction.

1 INTRODUCTION

The weights of a large language model can occupy more memory than a single device can hold. Post-training compression reduces this storage requirement using a frozen checkpoint and a small set of representative text, called calibration data, without retraining the model. Two common approaches are pruning, which sets selected weights to zero, and quantisation, which replaces weight values with representatives from a smaller set (Frantar & Alistarh, 2023; Frantar et al., 2023). Both change the outputs of the model's layers and can reduce accuracy, especially at high compression rates.

One way to recover accuracy is to add a small approximation of the error that compression leaves behind. Instead of replacing a layer's weights entirely with a low-rank matrix, we keep the compressed weights and add a correction in the form of two thin matrices. During inference, the layer adds the outputs of these two terms. EoRA applies the correction using inputs observed during calibration, so its limited rank is allocated to errors that affect layer outputs (Liu et al., 2026). This avoids fine-tuning, although the correction adds some storage and computation.

However, a pruner decides which weights to keep before knowing what the correction will recover, hence revisiting that choice could improve the combined representation. When the model is also quantised, we must decide whether to quantise at every revision or once at the start (Mozaffari et al., 2025) or the end. The quantiser algorithm matters too, since weighting its errors by input activations may duplicate what an activation-aware correction already does. This raises our question: how should pruning and quantisation be designed when a low-rank correction follows?

We answer this question with QUENCH, which designs pruning and quantisation around the correction (Figure 1). QUENCH first subtracts the preceding correction from the original weights before pruning again, then refits the correction. We call this loop 'Joint-SLR', for Joint Sparse and Low-rank Reconstruction. After this pruning loop, it quantises the kept weights once with elastic-bucket quantisation (EBQ), which fits a small codebook separately to each output channel, and refits the correction to the total error. This algorithm is developed by improving each part of the compression and correction steps, and we share the individual improvements too. We also analyse the trade-offs between activation-weighted compression versus activation-aware correction.

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.

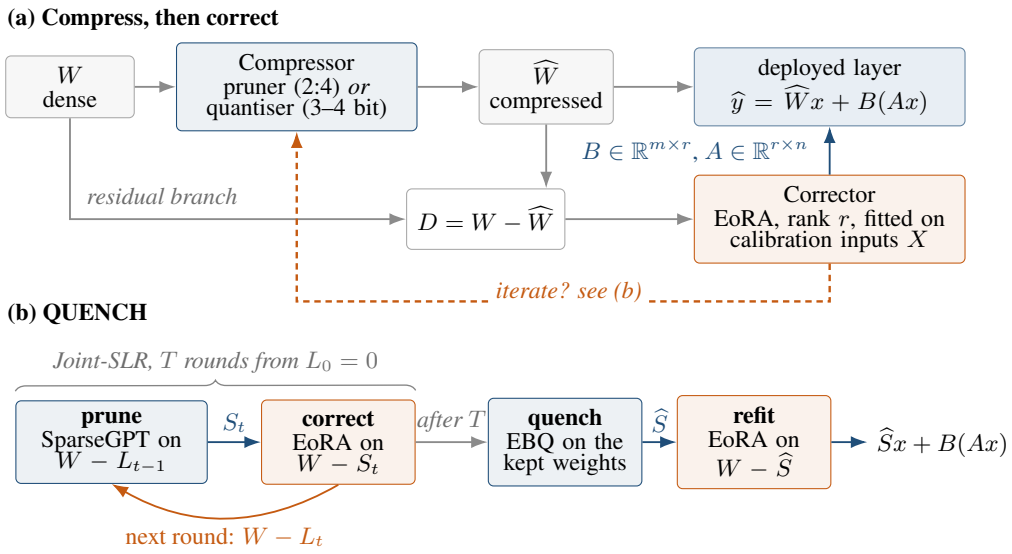


Figure 1: QUENCH and the compress-then-correct block it builds on. (a) A 2:4 pruner or 3–4 bit quantiser turns W into $\widehat{W}$, and EoRA fits a rank-r correction BA to the residual D on calibration inputs. The dashed arrow marks the question studied here: whether the correction should be fed back to the compressor. (b) QUENCH re-prunes $W - L_{t-1}$ and refits the correction for T rounds, then quantises the kept weights once with EBQ and refits the correction to the total error.

Our contributions are:

- **The QUENCH Algorithm.** A training-free pipeline that runs a pruning–correction loop, then quantises the kept weights once with EBQ and refits the correction. We also test a schedule wherein we quantise within every pruning iteration, and report the results for both techniques.
- **Pruning stage: Joint-SLR.** Joint-SLR alternates SparseGPT with EoRA while keeping the final correction rank fixed. We observe an increase in accuracy on the datasets tested.
- **Quantisation stage: EBQ with correction.** A six-condition ablation separates learned level placement, activation weighting and correction. We show how the unweighted EBQ becomes competitive with its weighted counterpart after EoRA correction and outperforms the local grouped-GPTQ comparator at three bits.

QUENCH introduces a schedule: iterate pruning and correction, quantise once, then refit the correction. It builds on SparseGPT, Lloyd quantisation and EoRA.

2 Related Work

Pruning. Pruning decides which weights can be removed to save memory with the least loss in accuracy. Weight-magnitude based pruning is the simplest criterion but ignores the inputs each weight multiplies. Wanda adds input-channel activation magnitudes to its importance score (Sun et al., 2024), and SparseGPT also uses activation statistics while adjusting the remaining weights to compensate for those removed (Frantar & Alistarh, 2023). We use SparseGPT under the 2:4 pattern, which keeps two of every four consecutive weights in a row; calibration data can materially affect this constrained problem (Ji et al., 2025). These methods fix the mask in a single pass, before any correction is fitted, whereas Joint-SLR revisits it after each correction.

Quantisation. Uniform quantisation places representative values on an equally spaced grid. GPTQ compensates for rounding errors by updating weights that have not yet been quantised, using calibration activations to estimate the effect on layer outputs (Frantar et al., 2023). Its grid can

cover a whole output channel or smaller groups of weights, with different storage costs. Learned codebooks instead adjust the locations of the representative values. SqueezeLLM uses Fisher-weighted clustering, LeanQuant uses inverse-Hessian-diagonal weights, and any4 derives activation-dependent centroid updates (Kim et al., 2024; Zhang & Shrivastava, 2025; Elhoushi & Johnson, 2025). EBQ implements scalar Lloyd quantisation (Lloyd, 1982). Its unweighted objective and uniform min–max initialisation also appear in LeanQuant when its weighting exponent is zero ($p = 0$), but LeanQuant uses them inside GPTQ-style sequential updates and without a low-rank correction. We test whether activation-second-moment weighting helps when a separate corrector follows.

Low-rank correction. A correction represents part of the compression residual using two small factors. ZeroQuant-V2 fits them with a plain SVD of the residual, and ApiQ trains them with gradients (Yao et al., 2024; Liao et al., 2024). LQER uses activation-based diagonal scaling, while EoRA and QERA develop activation-aware error reconstruction (Zhang et al., 2024; Liu et al., 2026; Zhang et al., 2025). LoftQ, LQ-LoRA and CALDERA alternate quantised and low-rank components (Li et al., 2024; Guo et al., 2024; Saha et al., 2024). Other approaches allocate part of the low-rank capacity before quantisation: SVDQuant does so for diffusion models, ODLRI assigns activation-sensitive weights to the low-rank component, and Preserve-Then-Quantize divides rank between preservation and reconstruction (Li et al., 2025; Cho et al., 2025; 2026). Scetbon & Hensman (2024) extend correction to activation quantisation; we quantise weights only and keep the corrector fixed to EoRA, so that compression is the only variable.

Joint compression and ordering. Other methods fit the compressed and low-rank parts jointly. OATS, HASSLE-free and 3BASiL optimise sparse-plus-low-rank representations (Zhang & Papyan, 2025; Makni et al., 2025a;b). TLRR refines sparse values under a fixed mask using data-free SVD (Shen et al., 2025); our loop permits SparseGPT to reselect support and uses activation-based correction. Kim et al. (2026) study the order in which pruning and quantisation are applied. LoftQ and SLiM are closest to QUENCH. LoftQ alternates quantisation with a low-rank fit, as Joint-SLR alternates pruning with the correction (Li et al., 2024). However, its fit is a plain SVD that only initialises adapters, which are then fine-tuned with gradients on each task. QUENCH is training-free: it needs only forward passes, fits an activation-aware correction and lets the pruner choose a new mask every round. SLiM also combines 2:4 sparsity, quantisation and a low-rank correction without training, but quantises before pruning once and scales the residual by mean absolute activations (Mozaffari et al., 2025). Unlike SLiM, QUENCH repeatedly revises the sparse weights through activation-aware correction, then learns codebooks for the final survivors. At equal rank on 3-bit LLaMA-3-8B, SLiM's correction reaches 25.91 MathQA, against 29.11 for EoRA, which QUENCH uses (Liu et al., 2026); our EBQ with EoRA reaches 37.25, outperforming SLiM's correction by over 11 points. Section 5.2 runs LoftQ's alternation without fine-tuning, and Table 5 lists SLiM's published results.

3 PRELIMINARIES

Consider one linear layer with weights $W \in \mathbb{R}^{m\times n}$ and input $x \in \mathbb{R}^n$. Its output is $y = Wx$, where n and m are the input and output widths. Compression replaces W with $\widehat{W}$, changing the output by $(W - \widehat{W})x$. We call $D = W - \widehat{W}$ the compression residual. The aim of correction is to recover part of Dx without storing all of D.

A rank-at-most-r correction writes $L = BA$, with $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$. The corrected layer computes

$$\widehat{y} = \widehat{W}x + B(Ax). \tag{1}$$

Instead of mn, the factors store $r(m + n)$ values, controlled by the rank r. The compressed matrix remains the base representation, while the correction supplies a second output path. Figure 1(a) illustrates the construction.

A truncated singular value decomposition (SVD) of D gives the best rank-r approximation in ordinary squared weight error. But equal weight errors need not produce equal output errors: some input directions occur more strongly than others. To account for this, let $X \in \mathbb{R}^{n\times N}$ collect N calibration input vectors as columns. EoRA fits the correction to the outputs on these inputs (Liu et al., 2026):

$$\min_{\mathrm{rank}(L)\le r} J(L), \qquad J(L) = \|WX - (\widehat{W} + L)X\|_{\mathrm{F}}^2 = \|(D - L)X\|_{\mathrm{F}}^2. \tag{2}$$

Here J is the total squared layer-output error and $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm, the square root of the sum of squared matrix entries. Calibration supplies representative inputs for fitting this objective; it does not require backpropagation through the model.

The input second-moment matrix $C = XX^\top$ contains the information needed for this fit. If C is positive definite, write $C = U\Lambda U^\top$ and $P = U\Lambda^{1/2}$, where the columns of U are eigenvectors and the diagonal entries of Λ are their eigenvalues. Multiplying the residual by P expresses it in these input directions and scales them by their strength. The correction is

$$L = \mathcal{R}_r(D; C) = [DP]_r P^{-1}, \tag{3}$$

where $[DP]_r$ denotes a best rank-at-most-r truncated SVD of DP, and $\mathcal{R}_r$ names this reconstruction operation. In simple words, transform the residual, approximate it at rank r, then map back. Appendix A gives the derivation and factor construction. In practice, C is accumulated as calibration inputs pass through the already-compressed earlier blocks, so each layer is corrected for the inputs it will receive; Appendix B shows the exact weighting and numerical precision.

This fixes the corrector used throughout QUENCH. The remaining choices concern the base matrix: whether to revise the sparse weights after correction, when to quantise them, and which quantisation levels to use.

4 THE QUENCH PIPELINE

4.1 ITERATIVE PRUNING AND CORRECTION

A sparse matrix stores selected weights and sets the others to zero. We use 2:4 sparsity: each group of four consecutive positions along a row retains two positions. The set of retained positions is its *support*. SparseGPT chooses the support using calibration inputs to estimate the effect of removal on layer outputs (Frantar & Alistarh, 2023). It processes weights sequentially and adjusts unprocessed values to compensate for the error already introduced. Thus it changes both the support and the retained values.

We write $\mathcal{P}_{2:4}(V)$ for this operation on a target matrix V, with its pruning calibration inputs implicit. A single pruning pass targets $V = W$, without accounting for a later low-rank correction.

The correction can also inform pruning. If L_{t-1} is the preceding correction, the next sparse matrix should represent the part of W that it leaves uncovered. Joint-SLR therefore starts with $L_0 = 0$ and applies

$$S_t = \mathcal{P}_{2:4}(W - L_{t-1}), \qquad L_t = \mathcal{R}_r(W - S_t; C_t), \qquad t = 1, \ldots, T. \tag{4}$$

Here t indexes rounds, T is their total number, S_t is the sparse base, and C_t contains the correction statistics for that round. The first round is the standard prune-then-correct pipeline. On later rounds, subtracting L_{t-1} changes the matrix presented to SparseGPT, allowing a different support and different retained values.

The second update fits $W - S_t$, because the target remains the original layer. It replaces L_{t-1} rather than adding another correction; the final rank stays r at every T. The implementation reloads the original checkpoint before each pruning and correction stage. It recomputes activations through sequentially modified blocks, so C_t can change between rounds. SparseGPT is also an approximate solver. These updates therefore do not imply descent on one fixed objective; related layer-input effects are studied by QEP (Arai & Ichikawa, 2025).

4.2 ELASTIC-BUCKET QUANTISATION

A b-bit code can select one of $K = 2^b$ values. For example, three bits allow eight representatives. An equally spaced grid may spend several of these on sparsely populated tails of a weight distribution. A learned codebook can move the representatives towards the more weights-dense part of the distribution.

EBQ fits a separate codebook $c_{i1}, \ldots, c_{iK}$ to each row i of W. Starting from an evenly spaced grid between the row's minimum and maximum weights, it alternates nearest-level assignment and

centroid updates:

$$z_{ij} = \arg\min_k |w_{ij} - c_{ik}|^2, \qquad c_{ik} \leftarrow \frac{\sum_{j:z_{ij}=k} s_j w_{ij}}{\sum_{j:z_{ij}=k} s_j}. \tag{5}$$

Here w_{ij} is a weight, z_{ij} selects its codebook entry, and $s_j > 0$ is the importance assigned to input channel j. We use $s_j = 1$ for unweighted codebooks or mean squared input activation for the weighted variant. Empty clusters keep their previous value. Fifteen Lloyd updates (a sufficient number across most of our experiments) are followed by a final assignment. Algorithm 1 then fits the correction to the resulting residual.

Algorithm 1 EBQ with low-rank correction, for one layer

Require: Original weights W, bit width b, rank r, importance weights s_j

Initialise 2^b evenly spaced representatives per row of W
for $\ell = 1, \ldots, 15$ **do**
    Assign each weight to its nearest representative using Equation 5
    Update occupied representatives using Equation 5; keep empty ones
end for
Assign once more; set $\widehat{W}_{ij} = c_{i,z_{ij}}$
Collect correction statistics C during the sequential correction pass
Factor $\mathcal{R}_r(W - \widehat{W}; C)$ as BA using Appendix A
return $\widehat{W}, B, A$, representing the layer in Equation 1

Our uniform control uses per-row min–max round-to-nearest (RTN). It omits GPTQ's sequential error compensation and isolates level placement within the ablation. GPTQ is evaluated separately at channel-wise and group-128 granularity. These are distinct comparators with different metadata costs.

4.3 Combining pruning and quantisation

Pruning reduces how many values the base must store, while quantisation reduces the bits used for those values. To combine them, QUENCH first completes the pruning–correction loop, then quantises the final sparse matrix once and refits the correction. We name it after the metalworking step that fixes a shape once, when forging is complete. To our knowledge, no prior method alternates mask selection with an activation-aware correction and then quantises the final support once with learned codebooks. Figure 1(b) and Algorithm 2 show both phases.

Algorithm 2 QUENCH: Joint-SLR, then one quantisation and a final correction

Require: Original weights W, rounds T, bit width b, rank r

$L \leftarrow 0$
for $t = 1, \ldots, T$ **do** ▷ Phase 1: Joint-SLR
    $S \leftarrow \mathcal{P}_{2:4}(W - L)$
    Collect correction statistics C_t during the sequential correction pass
    $L \leftarrow \mathcal{R}_r(W - S; C_t)$
end for
    For pruning alone, return S and the factors of L here.
$\widehat{S} \leftarrow \mathcal{Q}_b(S)$ ▷ Phase 2, quench: unweighted EBQ on retained values only
Collect final correction statistics C_{final}
Factor $\mathcal{R}_r(W - \widehat{S}; C_{\text{final}})$ as BA
return $\widehat{S}, B, A$

The quantiser $\mathcal{Q}_b$ fits unweighted codebooks only to retained values; pruned positions are excluded and explicitly reset to zero. Quantisation introduces a new error, so the last correction is refitted to the total residual $W - \widehat{S}$, where $\widehat{S}$ is the quantised sparse base. The correction produced within the loop is replaced. The sparse base retains its support constraint, although adding the dense low-rank correction need not preserve sparsity of the combined matrix.

The algorithms state the updates for one matrix. In the model, each compression stage and each correction stage runs across the transformer blocks in order, with separate calibration passes. The final quantisation adds one correction pass to the T fits made during pruning. Section 5.4 tests this choice against quantising within every round.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We compress the 224 linear projections of LLaMA-3-8B and LLaMA-2-7B (Grattafiori et al., 2024; Touvron et al., 2023), keeping embeddings, the output head and normalisation parameters at checkpoint precision. We report raw zero-shot ARC-Challenge (ARC-C) and MathQA accuracy, five-shot GSM8K accuracy (LLaMA-3 only; Appendix B) and WikiText2 test perplexity (PPL) (Clark et al., 2018; Amini et al., 2019; Cobbe et al., 2021; Merity et al., 2017). Compression is calibrated on 128 WikiText2 training sequences of 2,048 tokens. Each task then receives its own correction, fitted on 64 sequences that mix task training examples with C4 (Raffel et al., 2020); PPL rows use 128 WikiText2 sequences. Unless stated, the correction rank is 128 and results use one seed; the main 3-bit configuration has three. EoRA is the corrector throughout, as it outperforms residual SVD and activation scaling in a control experiment (Appendix C).

5.2 ITERATIVE PRUNING

Joint-SLR tests whether the sparse base benefits from being revised after correction. Its one-round task scores match our single-pass EoRA run (Table 6). Five rounds improve LLaMA-3 ARC-C by 4.86 points and PPL from 10.9058 to 9.9227 (Table 1), with no increase in final rank. Task improvement is not monotone: ARC-C dips at round three, while MathQA peaks there. At $T = 5$, Joint-SLR exceeds every published correction in Table 1 on all four metrics.

Table 1: Iterating compression and correction on LLaMA-3-8B, rank 128. Blocks: published results, Joint-SLR ($T = 1$ is the single pass), and the same loop with quantisation instead of pruning, which differs in storage. †Reported by Liu et al. (2026) for the same tasks and rank, with a similar calibration mix.

Configuration	ARC-C ↑	MathQA ↑	GSM8K ↑	PPL ↓
Published results on the same 2:4 model†				
SparseGPT 2:4 (Frantar & Alistarh, 2023)	30.11	26.43	2.12	12.32
+ ZeroQuant-V2 (Yao et al., 2024)	31.99	26.49	2.96	11.31
+ Act-S (Yuan et al., 2023)	31.74	26.73	3.26	11.32
+ ApiQ (Liao et al., 2024), trained	34.21	28.77	14.55	11.08
+ EoRA (Liu et al., 2026)	34.64	29.91	13.95	11.07
Ours: Joint-SLR, 2:4 SparseGPT alternated with EoRA				
Joint-SLR, $T = 1$	35.15	29.85	13.65	10.91
Joint-SLR, $T = 2$	37.80	30.42	14.18	10.43
Joint-SLR, $T = 3$	37.54	**31.46**	14.71	10.17
Joint-SLR, $T = 5$	**40.02**	30.79	**15.39**	**9.92**
Quantisation instead of pruning (3-bit GPTQ, LoftQ-style): no consistent gain				
Same loop, $T = 1$	32.25	27.47	11.83	10.62
Same loop, $T = 2$	32.00	28.14	13.19	10.56
Same loop, $T = 3$	32.08	26.77	12.66	10.62
Same loop, $T = 5$	33.28	27.14	13.19	10.74

Repeating quantisation gives a different response. The bottom block of Table 1 runs LoftQ's alternation (Li et al., 2024) without its fine-tuning, with channel-wise 3-bit GPTQ and EoRA in place of NormalFloat quantisation and a plain SVD. Over five rounds, PPL moves from 10.6166 to 10.7403; ARC-C improves while MathQA falls. This motivates quantising once, outside the pruning loop, as QUENCH does. It does not show that iterative quantisation fails generally.

5.3 ELASTIC-BUCKET QUANTISATION

We evaluate EBQ on dense weights, where the quantiser's effect is isolated. The six conditions in Table 2 and Figure 3 separate level placement, weighting and correction. Without correction, activation weighting improves EBQ codebooks by 12.29 ARC-C points and reduces PPL from 22.81 to 8.77. With correction, the unweighted variant is 0.60 points higher on ARC-C and 1.31 on MathQA, while the weighted variant is 0.15 points higher on GSM8K. The ARC-C and GSM8K gaps correspond to seven and two correctly answered items, respectively. There is no consistent additional benefit from weighting after correction in this setting.

Table 2: Attribution grid for 3-bit LLaMA-3-8B, without and with rank-128 EoRA correction. Each column uses a correction fitted on that task's own calibration data.

	ARC-C ↑		MathQA ↑		GSM8K ↑		PPL ↓	
Quantiser	none	EoRA	none	EoRA	none	EoRA	none	EoRA
Uniform RTN	19.11	30.46	20.64	25.83	0.00	11.52	36753	11.44
EBQ, unweighted	29.52	**47.44**	26.03	**37.25**	0.15	27.52	22.81	**7.24**
EBQ, activation-weighted	41.81	46.84	33.07	35.95	10.77	**27.67**	8.77	7.26

Learned placement still matters when correction is present. Unweighted EBQ exceeds corrected uniform RTN by 16.98 ARC-C points. Against the stronger local group-128 GPTQ comparator, its gain is 2.99 points using the three-seed mean (Table 3), which retains 94.01% of dense ARC-C accuracy. Published corrections of the same 3-bit GPTQ model reach at most 31.74 ARC-C, against 47.44, at a similar budget (about 3.77 against 3.79 bits per weight). Across eleven of them, including LQER, QERA, CALDERA, SLiM and OATS, the best MathQA is 29.11, against 37.25 (Appendix C). The weighting ablation tests input second moments only. The Fisher, inverse-Hessian and any4 weighting rules require further testing.

Table 3: LLaMA-3-8B, 3-bit weights, rank-128 correction. †Reported by Liu et al. (2026); ApiQ trains its factors with gradients. Our channel-wise GPTQ row re-runs the reported EoRA pipeline as a check on our evaluation.

Configuration	ARC-C ↑	MathQA ↑	GSM8K ↑	PPL ↓
Dense (16-bit)	49.83	40.30	35.94	6.14
Published results on the same 3-bit channel-wise GPTQ model†				
GPTQ (Frantar et al., 2023)	20.90	22.37	0.45	15.64
+ ZeroQuant-V2 (Yao et al., 2024)	30.02	26.43	3.79	10.24
+ Act-S (Yuan et al., 2023)	31.28	25.42	4.09	10.19
+ ApiQ (Liao et al., 2024), trained	30.46	26.86	10.79	10.41
+ EoRA (Liu et al., 2026)	31.74	29.11	11.90	10.06
Ours, all with EoRA				
GPTQ, channel-wise (re-run)	32.25	27.47	11.83	10.62
GPTQ, group 128	43.86	33.87	26.54	7.50
EBQ, seed 0	**47.44**	**37.25**	**27.52**	**7.24**
EBQ, three-seed mean	46.84 ± 0.56	37.19 ± 0.15	27.45 ± 0.08	7.25 ± 0.01

The advantage of learned placement shrinks as more bits become available: at four bits, EBQ is 1.71 ARC-C points above corrected uniform RTN (48.89% against 47.18%), compared with 16.98 points at three bits. Raising the rank from 64 to 256 steadily lowers 3-bit PPL, but ARC-C peaks at rank 128 (Appendix C).

Representation cost. Each compressed weight is stored as a b-bit index into its row's codebook, each row keeps its 2^b codebook values in 16 bits, and the correction adds two 16-bit factors. For $W \in \mathbb{R}^{m \times n}$, the ideal packed size is therefore

$$S(W) = \underbrace{bmn}_{\text{indices}} + \underbrace{16m2^b}_{\text{row codebooks}} + \underbrace{16r(m+n)}_{\text{correction factors}} \quad \text{bits.} \tag{6}$$

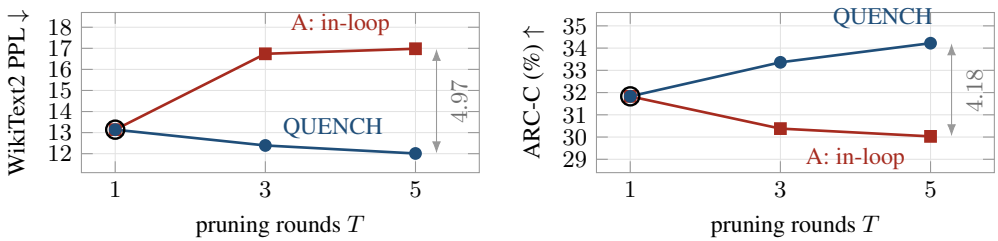


Figure 2: Quantising inside the pruning loop (A) against QUENCH on LLaMA-3-8B (2:4 support, 3-bit retained values, rank 128). At $T = 1$ the two are the same computation (hollow marker). After five rounds, QUENCH has improved by 1.14 perplexity and 2.39 ARC-C points, while A has worsened by 3.83 and 1.79.

At $b = 3$ and $r = 128$, this averages 3.79 bits per weight over LLaMA-3's compressed projections: 3 bits for the indices, 0.03 for the codebooks and 0.77 for the correction. The projections shrink from 14.0 GB at 16 bits to 3.3 GB, and the whole model from 16.1 GB to about 5.4 GB, with the remaining parameters kept at 16 bits. These are sizes of an ideal packed format; inference memory and speed depend on packed kernels, which are outside the scope of this study.

We also analyse a tradeoff wherein SqueezeLLM (Kim et al., 2024) remains more bit-efficient: on LLaMA-2-7B, it reaches 5.96 perplexity with 3.24 bits per weight, whereas EBQ with correction needs 3.42 bits for a similar 5.91, mostly to store the correction. In exchange, EBQ is cheaper to build. Its codebooks are fitted to the weights alone and its correction to forward-pass activations, while SqueezeLLM's Fisher sensitivities need a backward pass through the whole model.

5.4 Combining pruning and quantisation

We compare QUENCH with an alternative (A) that quantises the retained values on every pruning round, before fitting its correction. Both use 2:4 support, 3-bit unweighted codebooks and rank 128. Their final representation costs agree. Their fitting costs differ: A makes T quantisation and T correction passes; QUENCH makes one quantisation and $T + 1$ correction passes.

At $T = 1$, the final computations are algebraically equivalent because QUENCH's preliminary correction is discarded. The one-round task scores are therefore shared references. At $T = 5$, QUENCH gives PPL 12.0056 against A's 16.9794 (Table 4, Figure 2). QUENCH's ARC-C rises from 31.83% to 34.22% and GSM8K from 6.97% to 7.73%, while A falls to 30.03% and 6.14%. QUENCH leads on every metric at three and five rounds. At five rounds it also beats every published correction of 2:4 with 4-bit GPTQ on ARC-C and PPL with three-bit values, though ApiQ and EoRA keep higher GSM8K.

Table 4: Combined compression on LLaMA-3-8B. A quantises within every round; QUENCH quantises once after pruning. Published corrections at 2:4 with 4-bit GPTQ reach ARC-C/GSM8K/PPL of 29.35/1.59/12.84 (ZeroQuant-V2), 27.90/1.90/12.99 (Act-S), 30.71/11.06/12.77 (ApiQ) and 31.22/10.16/12.60 (EoRA) (Liu et al., 2026).

	A: quantise in loop			QUENCH		
Rounds	ARC-C ↑	GSM8K ↑	PPL ↓	ARC-C ↑	GSM8K ↑	PPL ↓
1	31.83	6.97	13.14	31.83	6.97	13.14
3	30.38	6.52	16.74	33.36	7.43	12.39
5	30.03	6.14	16.98	**34.22**	**7.73**	**12.01**

Checks on LLaMA-2-7B. On LLaMA-2-7B (Table 5), from one to five rounds, Joint-SLR improves ARC-C from 32.34% to 33.28% and PPL from 7.98 to 7.31, and QUENCH from 30.89% to 32.00% and from 9.29 to 8.26, so the iterative gains hold on a second checkpoint. A and the weighting ablation are reported for LLaMA-3 only.

Table 5: LLaMA-2-7B, rank 128 unless stated; each task uses its own correction. †Reported by Liu et al. (2026) with channel-wise GPTQ; their combined row uses 4-bit weights, QUENCH 3-bit. Appendix C lists all published corrections. Bold: best per column within each block.

Configuration	ARC-C ↑	MathQA ↑	PPL ↓
EoRA (Liu et al., 2026), 2:4 SparseGPT†	32.67	25.59	7.97
Joint-SLR, $T = 1$	32.34	26.60	7.98
Joint-SLR, $T = 5$	**33.28**	**26.73**	**7.31**
EoRA, 3-bit GPTQ†	35.83	25.79	6.84
EBQ + EoRA, 3-bit	**37.20**	**26.33**	**5.86**
EoRA, 4-bit GPTQ†	38.05	**27.13**	5.68
EBQ + EoRA, 4-bit	**39.33**	26.90	**5.60**
2:4 with 4-bit values, reported by Mozaffari et al. (2025); PPL only			
SparseGPT + GPTQ, group 128	–	–	12.07
Wanda + RTN, group 128	–	–	14.36
SLiM (Mozaffari et al., 2025), rank ≈410	–	–	**7.77**
EoRA, 2:4 + 4-bit GPTQ†	31.14	**25.39**	**8.24**
QUENCH, 3-bit, $T = 1$	30.89	25.13	9.29
QUENCH, 3-bit, $T = 5$	**32.00**	25.33	8.26

At three bits, QUENCH's 8.26 PPL is far below the 12.07 to 14.36 of the one-shot 2:4 baselines that SLiM reports at four bits, while SLiM itself reaches 7.77 with a correction about three times larger than ours. QUENCH's final step can also follow other pruners, such as 3BASiL (Makni et al., 2025b).

6 Discussion and conclusion

We have formulated and tested QUENCH to design pruning and quantisation around a low-rank correction. On LLaMA-3-8B with 2:4 sparsity and 3-bit values, it improves on the perplexity and ARC-C that EoRA reports with 4-bit weights, without training. The results point to one principle: when a correction follows compression, how the stages are composed matters as much as how each performs alone. At about a third of the 16-bit size, QUENCH suits memory-constrained devices.

Iteration lets the sparse base and the correction adapt to each other: once the correction covers part of a layer, the next pruning round can spend its retained weights on what the correction leaves uncovered. Quantising inside the loop disturbs this exchange. In the in-loop schedule A, each correction also absorbs the rounding errors of that round's codebooks, and the next round's re-quantisation replaces those errors with new ones. QUENCH improves all reported metrics across the tested rounds, while A worsens. Changing codebooks may make the previous correction less useful for the next pruning step. Tracking mask and codebook changes, with fixed-mask tests and matched correction-pass counts, would test this explanation.

Within this design, the quantiser and the correction divide the work: EBQ codebooks represent the distribution of weight values, while EoRA reconstructs the errors that matter on calibration inputs. Activation weighting greatly improves a standalone codebook, likely because nothing else accounts for which input channels matter. After correction, EoRA fills this role: it weights errors by the full second-moment matrix of its calibration inputs, whereas the codebook weighting uses only per-channel second moments. Weighting then adds no consistent gain, so QUENCH keeps the simpler unweighted codebook.

Our experiments cover two LLaMA models, each with task-specific corrections. Natural next steps are other model families, other sensitivity measures for the codebook, a single correction shared across tasks, and packed kernels that turn the computed storage savings into measured memory and speed gains.

AI USE STATEMENT

We used generative AI tools to correct grammar and improve readability for parts of the manuscript from existing research, and prepare tables and figures. The authors reviewed all AI-assisted content and take full responsibility for the paper and any accompanying materials.

ETHICS STATEMENT

The work uses existing language-model checkpoints and public benchmark datasets, subject to their original terms. Compression can alter model behaviour beyond the task accuracies measured here; this study does not establish preserved safety or fairness, and it does not measure energy savings.

REPRODUCIBILITY STATEMENT

Sections 3–4 and Algorithms 1 and 2 specify the method. Section 5 and Appendix B give the models, calibration data, hyperparameters, evaluation protocol and software versions, and Appendix A derives the correction. Our values come from single runs, except the three-seed mean in Table 3; the calibration seeds are listed in Appendix C. We will release the code upon acceptance.

REFERENCES

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019. URL https://aclanthology.org/N19-1245/.

Yamato Arai and Yuma Ichikawa. Quantization error propagation: Revisiting layer-wise post-training quantization. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/df2034a516cbd617a96492cc476276c9-Abstract-Conference.html.

Yoonjun Cho, Soeun Kim, Dongjae Jeon, Kyelim Lee, Beomsoo Lee, and Albert No. Assigning distinct roles to quantized and low-rank matrices toward optimal weight decomposition. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. URL https://aclanthology.org/2025.findings-acl.746/.

Yoonjun Cho, Dongjae Jeon, Soeun Kim, Moongyu Jeon, and Albert No. Preserve-Then-Quantize: Balancing rank budgets for quantization error reconstruction in LLMs. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.02001.

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. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

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

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. URL https://doi.org/10.1007/BF02288367.

Mostafa Elhoushi and Jeff Johnson. any4: Learned 4-bit numeric representation for LLMs. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/elhoushi25a.html.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2301.00774.

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*, 2023. URL https://arxiv.org/abs/2210.17323.

Leo Gao, Jonathan Tow, Stella Biderman, et al. EleutherAI/lm-evaluation-harness: v0.3.0, 2022. URL https://doi.org/10.5281/zenodo.7413426.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Han Guo, Philip Greengard, Eric P. Xing, and Yoon Kim. LQ-LoRA: Low-rank plus quantized matrix decomposition for efficient language model finetuning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xw29VvOMmU.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2207.00112.

Yixin Ji, Yang Xiang, Juntao Li, Qingrong Xia, Ping Li, Xinyu Duan, Zhefeng Wang, and Min Zhang. Beware of calibration data for pruning large language models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/2ede933e10afa991a10b6f36b6522129-Abstract-Conference.html.

Minjun Kim, Jaehyeon Choi, Hyunwoo Yang, Jongjin Kim, Jinho Song, and U Kang. Prune-then-quantize or quantize-then-prune? Understanding the impact of compression order in joint model compression. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/578447217543a987c3a0d5b8922f3591-Abstract-Conference.html.

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*, 2024. URL https://arxiv.org/abs/2306.07629.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. SVDQuant: Absorbing outliers by low-rank components for 4-bit diffusion models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2411.05007.

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. LoftQ: LoRA-fine-tuning-aware quantization for large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/39ec972afab01e0d8ddc6834a9d12ac1-Abstract-Conference.html.

Baohao Liao, Christian Herold, Shahram Khadivi, and Christof Monz. ApiQ: Finetuning of 2-bit quantized large language model. In *Conference on Empirical Methods in Natural Language Processing*, 2024. URL https://arxiv.org/abs/2402.05147.

Shih-Yang Liu, Maksim Khadkevich, Nai Chit Fung, Charbel Sakr, Chao-Han Huck Yang, Chien-Yi Wang, Saurav Muralidharan, Hongxu Yin, Kwang-Ting Cheng, Jan Kautz, Yu-Chiang Frank Wang, Pavlo Molchanov, and Min-Hung Chen. EoRA: Fine-tuning-free compensation for compressed LLM with eigenspace low-rank approximation. *arXiv preprint arXiv:2410.21271v6*, 2026. URL https://arxiv.org/abs/2410.21271v6.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28 (2):129–137, 1982. URL https://doi.org/10.1109/TIT.1982.1056489.

Mehdi Makni, Kayhan Behdin, Zheng Xu, Natalia Ponomareva, and Rahul Mazumder. A unified framework for sparse plus low-rank matrix decomposition for LLMs. In *Conference on Parsimony and Learning*, volume 280 of *Proceedings of Machine Learning Research*, pp. 484–499, 2025a. URL https://proceedings.mlr.press/v280/makni25a.html.

Mehdi Makni, Xiang Meng, and Rahul Mazumder. 3BASiL: An algorithmic framework for sparse plus low-rank compression of LLMs. In *Advances in Neural Information Processing Systems*, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/fe498359454d826def8a847fad753dc2-Abstract-Conference.html.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1609.07843.

Mohammad Mozaffari, Amir Yazdanbakhsh, and Maryam Mehri Dehnavi. SLiM: One-shot quantization and sparsity with low-rank approximation for LLM weight compression. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/mozaffari25a.html.

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. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea J. Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.18886.

Meyer Scetbon and James Hensman. Low-rank correction for quantized LLMs. *arXiv preprint arXiv:2412.07902*, 2024. URL https://arxiv.org/abs/2412.07902.

Li Shen, Anke Tang, Yong Luo, Tao Sun, Han Hu, and Xiaochun Cao. Targeted low-rank refinement: Enhancing sparse language models with precision. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/shen25e.html.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.11695.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL https://arxiv.org/abs/2307.09288.

Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, and Yuxiong He. Exploring post-training quantization in LLMs from comprehensive study to low rank compensation. In *AAAI Conference on Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2303.08302.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023. URL https://arxiv.org/abs/2312.05821.

Cheng Zhang, Jianyi Cheng, George A. Constantinides, and Yiren Zhao. LQER: Low-rank quantization error reconstruction for LLMs. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.02446.

Cheng Zhang, Jeffrey T. H. Wong, Can Xiao, George A. Constantinides, and Yiren Zhao. QERA: an analytical framework for quantization error reconstruction. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.06040.

Stephen Zhang and Vardan Papyan. OATS: Outlier-aware pruning through sparse and low rank decomposition. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=DLDuVbxORA.

Tianyi Zhang and Anshumali Shrivastava. LeanQuant: Accurate and scalable large language model quantization with loss-error-aware grid. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/57ccc284de6f060c8dcde8f9352f70a5-Abstract-Conference.html.

A Derivation of the weighted correction

Let $D = W - \widehat{W}$ and $C = XX^\top = PP^\top$ with $P = U\Lambda^{1/2}$ invertible, and let tr denote the trace. Then

$$\|(D - L)X\|_{\mathrm{F}}^2 = \mathrm{tr}[(D - L)C(D - L)^\top] = \|DP - LP\|_{\mathrm{F}}^2. \tag{7}$$

Since P is invertible, $Z = LP$ ranges over all rank-at-most-r matrices as L does, so the Eckart–Young theorem (Eckart & Young, 1936) gives $Z^\star = [DP]_r$ and $L^\star = [DP]_r P^{-1}$, with minimum loss $\sum_{j>r} \sigma_j(DP)^2$. Writing $[DP]_r = U_{D,r}\Sigma_{D,r}V_{D,r}^\top$, the factors are $B = U_{D,r}\Sigma_{D,r}^{1/2}$ and $A = \Sigma_{D,r}^{1/2}V_{D,r}^\top P^{-1}$. The derivation assumes C is positive definite. The implementation replaces negative eigenvalues with the smallest positive eigenvalue but leaves exact zeros unchanged, so it does not guarantee invertibility.

B Experimental details

Calibration. Embeddings, the output head and normalisation parameters keep checkpoint precision: bfloat16 for LLaMA-3 and float16 for LLaMA-2. Compression windows are sampled from the `wikitext-2-raw-v1` training stream (Merity et al., 2017). Each task correction uses 64 sequences of training questions with their correct answers or rationales, filled with `sliuau/c4-train` windows (Raffel et al., 2020): 20 ARC-Easy, 10 ARC-Challenge and 34 C4 for ARC; 20 task and 44 C4 for MathQA; and 32 of each for GSM8K. Each task column is a separately corrected model. Compression takes seed s and correction seed $s + 1$; task examples are assembled sequentially, so changing the seed does not resample the full calibration set. Differences are computed before rounding and are not significance tests.

Evaluation. We use lm-evaluation-harness 0.3.0 (Gao et al., 2022). ARC-C and MathQA use unnormalised multiple-choice accuracy; GSM8K uses five demonstrations and numerical-answer extraction after `####`. The LLaMA-3 wrapper allows 256 new tokens and stops at the requested token, whereas the LLaMA-2 wrapper ignores the stop token and caps the total length at 1,500 tokens, so GSM8K is reported for LLaMA-3 only. WikiText2 PPL uses non-overlapping 2,048-token windows.

Numerics and hyperparameters. Before each Gram contribution, the second-moment accumulator is multiplied by $N_s/(N_s + 1)$ and the contribution divided by N_s is added, where N_s is the number of calibration sequences; this weights later sequences more heavily than an arithmetic mean. Eigendecomposition uses float64, the scaled SVD float32, and the factors checkpoint precision; dense products stored between rounds use bfloat16. SparseGPT uses block size 128 and damping 0.01. EBQ floors importance weights at 10^{-8} and runs fifteen float32 Lloyd updates; its GPU scatter-add is not bitwise deterministic. Each job used one 48 GB L40S with PyTorch 2.1.2, CUDA 12.1, transformers 4.36.0 and datasets 2.18.0.

C Additional results

Corrector comparison. With the same 2:4 pruned base, GSM8K accuracy is 1.52% without correction, 1.82% with residual SVD, 2.20% with mean-absolute-activation scaling (Act-S), and 13.65% with EoRA, which also leads with 4-bit quantisation and on ARC-C and MathQA (Table 6). Since Act-S uses mean absolute activations rather than second moments, this control supports the chosen corrector but does not isolate the contribution of off-diagonal covariance.

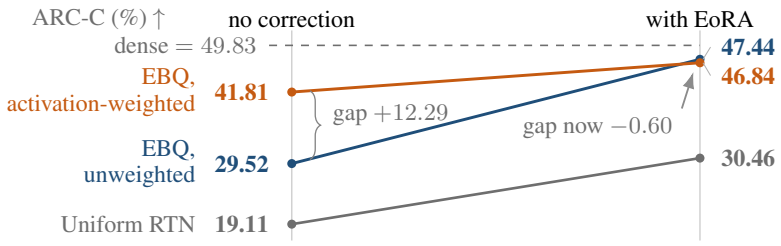


Figure 3: ARC-C of 3-bit LLaMA-3-8B for each quantiser, without (left) and with (right) rank-128 EoRA correction. Activation weighting leads unweighted EBQ by 12.29 points on its own but trails it by 0.60 points once corrected.

Table 6: Local corrector comparison on LLaMA-3-8B. SVD, Act-S and EoRA use rank 128. Act-S uses mean absolute activation scaling. The eigenspace and scaling variants use the same task-calibration recipe. For 2:4 + 4-bit, SparseGPT prunes and quantises (4-bit, per channel) in one pass.

Compression	Corrector	ARC-C ↑	MathQA ↑	GSM8K ↑
2:4	None	28.24	25.56	1.52
2:4	SVD	32.76	26.37	1.82
2:4	Act-S	31.23	26.83	2.20
2:4	EoRA	35.15	29.85	13.65
2:4 + 4-bit	None	18.86	20.80	0.00
2:4 + 4-bit	SVD	29.35	25.73	1.74
2:4 + 4-bit	Act-S	29.18	24.52	2.58
2:4 + 4-bit	EoRA	31.31	28.24	11.22

Published baselines. Table 7 collects the eleven low-rank corrections that Liu et al. (2026) evaluate on GPTQ-quantised LLaMA-3-8B at rank 128 with identical calibration data, and Table 8 their LLaMA-2-7B comparisons. Each published row corrects the same GPTQ or SparseGPT weights; our rows change the base, with EBQ or Joint-SLR, at the same rank.

Table 7: MathQA accuracy of low-rank corrections for GPTQ-quantised LLaMA-3-8B at rank 128, as reported by Liu et al. (2026), where CALDERA is adapted with fixed quantised weights. EBQ + EoRA replaces the GPTQ base (seed zero) at 3.79 and 4.82 bits per weight, against about 3.77 and 4.77 for GPTQ with the same correction.

Correction	3-bit ↑	4-bit ↑
None	22.37	34.07
FWSVD (Hsu et al., 2022)	26.30	35.64
ZeroQuant-V2 (Yao et al., 2024)	26.43	36.51
Act-S (Yuan et al., 2023)	25.42	35.84
ApiQ (Liao et al., 2024), trained	26.86	36.18
LQER (Zhang et al., 2024)	25.60	35.46
LRC (Scetbon & Hensman, 2024)	28.64	36.40
CALDERA (Saha et al., 2024)	28.10	36.70
QERA (Zhang et al., 2025)	25.32	35.90
SLiM (Mozaffari et al., 2025)	25.91	35.90
OATS (Zhang & Papyan, 2025)	25.30	36.01
EoRA (Liu et al., 2026)	29.11	37.21
EBQ + EoRA (ours)	**37.25**	**38.96**

Table 8: Published corrections of LLaMA-2-7B at rank 128, reported by Liu et al. (2026), with our corresponding rows. ZeroQuant-V2 fits a plain SVD, Act-S scales by mean absolute activations and ApiQ trains its factors. Bold: best per column within each block.

Compression	Correction	ARC-C ↑	MathQA ↑	PPL ↓
2:4 SparseGPT	ZeroQuant-V2	30.54	24.89	8.15
	Act-S	30.20	25.09	8.22
	ApiQ, trained	32.67	26.36	8.03
	EoRA	32.67	25.59	7.97
Joint-SLR, $T = 5$ (ours)	EoRA	**33.28**	**26.73**	**7.31**
3-bit GPTQ	ZeroQuant-V2	34.47	23.90	6.84
	Act-S	32.67	25.02	6.86
	ApiQ, trained	33.70	26.06	6.86
	EoRA	35.83	25.79	6.84
3-bit EBQ (ours)	EoRA	**37.20**	**26.33**	**5.86**
4-bit GPTQ	ZeroQuant-V2	37.62	27.06	5.68
	Act-S	**39.84**	**27.50**	5.68
	ApiQ, trained	39.59	27.00	5.68
	EoRA	38.05	27.13	5.68
4-bit EBQ (ours)	EoRA	39.33	26.90	**5.60**
2:4 + 4-bit GPTQ	ZeroQuant-V2	29.94	24.42	8.42
	Act-S	28.92	24.05	8.24
	ApiQ, trained	30.63	24.12	**8.03**
	EoRA	31.14	**25.39**	8.24
QUENCH, 3-bit, $T = 5$ (ours)	EoRA	**32.00**	25.33	8.26

Table 9: All calibration seeds for 3-bit unweighted EBQ with rank-128 correction on LLaMA-3-8B. Seed zero is the component-ablation run. The three-seed ARC-C mean (549 of 1,172 items) coincides with the weighted seed-zero score.

Seed	ARC-C ↑	MathQA ↑	GSM8K ↑	PPL ↓
0	47.44	37.25	27.52	7.24
1	46.33	37.29	27.37	7.26
2	46.76	37.02	27.45	7.25

Table 10: Rank and bit-width sweeps for LLaMA-3-8B. Effective bits include 16-bit factors and codebooks over affected linear weights. All rows use seed zero.

Bits	Rank	Eff. bits	ARC-C ↑	MathQA ↑	GSM8K ↑	PPL ↓
3	64	3.41	45.73	36.25	26.31	7.42
3	128	3.79	47.44	37.25	27.52	7.24
3	256	4.56	46.84	37.86	31.39	7.05
4	128	4.82	48.89	38.96	32.37	6.54