QSLR: A QUANTIZED + SPARSE + LOW-RANK DECOMPOSITION OF LLM WEIGHTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training Quantization (PTQ) of Large Language Models (LLMs) is a popular approach to reduce inference costs but can be restrictive under aggressive compression constraints. Hence, recent methods augment quantized weights with (i) sparse corrections $\mathbf{Q}+\mathbf{S}$ to capture outliers (e.g., SpQR is a specialized solver for the quantized-plus-sparse format) or (ii) low-rank corrections $\mathbf{Q}+\mathbf{L}$ to capture shared structure. We combine both in a unified quantized-plus-sparse-plus-low-rank representation, $\mathbf{W} \approx \mathbf{Q}+\mathbf{S}+\mathbf{L}$, and propose an ADMM-based algorithm `QSLR` that fits all three components jointly. For each layer, we minimize the reconstruction error $\|\mathbf{X}(\mathbf{W}-\mathbf{Q}-\mathbf{S}-\mathbf{L})\|_F^2$ subject to quantization, sparsity, and rank constraints, where $\mathbf{X}$ is the layer's input activation matrix. Our novel ADMM framework uses computationally efficient subroutines and is flexible to include different quantization formats. Extensive experiments demonstrate our joint optimization outperforms sequential fitting procedures. `QSLR` reduces the WikiText-2 perplexity of a Llama-3.2-1B from 16.79 to 14.82 compared to fitting SpQR followed by a low-rank correction, at the same compression budget. Additionally, we show that splitting the allocation between sparsity and low rank can improve performance over using either correction alone. For Qwen3-4B with (sym 2-bit) quantization at 2.47 bits per weight, using `QSLR` to fit all three allocations, the mixed representation achieves a WikiText-2 perplexity of 9.76, reducing the gap to the dense model by 30.9% relative to $\mathbf{Q}+\mathbf{S}$ and 35.2% relative to $\mathbf{Q}+\mathbf{L}$. Finally, we design GPU kernels that support $\mathbf{Q}+\mathbf{S}+\mathbf{L}$ and obtain important speedups over FP16 dense computation in matrix-multiplication and decoding benchmarks.

1 INTRODUCTION

Recent advances have expanded the capabilities of Large Language Models (LLMs) beyond conventional language generation to increasingly complex tasks, including resolving real-world software engineering problems (Jimenez et al., 2024), autonomously interacting with computer environments (Xie et al., 2024), and assisting scientific discovery through hypothesis generation and refinement (Tie et al., 2026). However, state-of-the-art LLMs (Yang et al., 2025; Team et al., 2026), often comprising billions of parameters, impose substantial computational and memory costs at inference time. These requirements hinder deployment in latency-sensitive settings and on resource-constrained hardware, motivating model compression methods that reduce inference costs while preserving model performance.

Quantization (Han et al., 2015; Frantar et al., 2023; Lin et al., 2024; Lucas et al., 2026), pruning (Frantar & Alistarh, 2023; Meng et al., 2024), and low-rank decomposition (Wang et al., 2025; Yuan et al., 2025) have emerged as successful approaches to reduce the cost of deploying LLMs while retaining performance. These directions have advanced largely independently, each accumulating a mature literature of specialized solvers. More recent work departs from this norm by *combining* two of the three primitives. SpQR (Dettmers et al., 2023) pairs quantization with sparsity $\mathbf{Q}+\mathbf{S}$, isolating the small fraction of outlier weights that dominate quantization error and keeping them in high precision while the rest are compressed to 3–4 bits, enabling near-lossless compression at low bit-widths. From the quantized-plus-low-rank perspective $\mathbf{Q}+\mathbf{L}$, CALDERA (Saha et al., 2024), LoftQ (Li et al., 2024), and LQ-LoRA (Guo et al., 2024) decompose each weight matrix as a quantized component plus a low-rank correction. Finally, OATS (Zhang & Papyan, 2025),

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.

HASSLE-free (Makni et al., 2025a) and 3BASiL (Makni et al., 2025b) target the sparse-plus-low-rank decomposition $\mathbf{S} + \mathbf{L}$. 3BASiL and HASSLE-free both minimize the layerwise reconstruction error under a prespecified sparsity pattern. The low-rank term provides additional flexibility in compression and absorbs the residual error from pure pruning.

In this paper, we introduce a novel quantized-plus-sparse-plus-low-rank representation $\mathbf{W} \approx \mathbf{Q} + \mathbf{S} + \mathbf{L}$. The quantized matrix $\mathbf{Q}$ provides a compact backbone of the pretrained weight, the sparse component $\mathbf{S}$ represents outlier entries, and the low-rank component $\mathbf{L}$ captures shared structure.

This representation raises two central research questions: How can we jointly fit $\mathbf{Q}$, $\mathbf{S}$, and $\mathbf{L}$? Additionally, at a fixed number of bits per weight (BPW), how should we divide the correction budget between sparsity and low rank for best performance?

A natural approach is to fit these components sequentially using existing mature compression methods. For example, we can augment SpQR, a specialized solver for $\mathbf{Q} + \mathbf{S}$, with an activation-aware low-rank correction (`wSVD`) fitted to its residual (Equation (5)), following the principle of EoRA (Liu et al., 2025). However, the quantized weights and sparse support are chosen without accounting for the low-rank correction that follows. Reversing the order has a similar limitation: the low-rank component is fitted without accounting for the subsequent quantization and pruning errors. These dependencies motivate a joint fitting procedure in which all three components adapt to one another under a common reconstruction objective.

In this paper, we propose `QSLR` that jointly fits the three components under explicit quantization, sparsity, and rank constraints based on the Alternating Direction Method of Multipliers (ADMM). We present the problem formulation and algorithm derivation in Section 3, with a fixed-grid convergence result for *scalar quantization* in Section B. In subsection 4.3, we demonstrate the gains obtained using `QSLR` over using sequential solvers on the residuals. In subsection 4.5, we compare the tradeoffs of using a sparse against a low-rank correction at fixed BPW and find that the mixed representation achieves the best performance among the tested allocations in many quantization settings. We complement our experimental studies with extensions to additional vector quantization formats like the E8 lattice codebook introduced in QuIP# (Tseng et al., 2024) as well as the effect of applying LoRA (Hu et al., 2022) finetuning for the low-rank correction, both of which are deferred to subsections C.4 and C.5. Finally, we study the effect of a non-uniform allocation of quantization, sparsity, and rank for *different layers* of an LLM under a *global* compression budget in subsection 4.6. Our **main contributions** can be summarized as follows.

1. **Joint quantized, sparse, and low-rank decomposition** We introduce `QSLR`, an ADMM-based procedure that jointly fits quantized, sparse, and low-rank components under a common layerwise reconstruction objective, with convergence guarantees for fixed scalar quantization grids. Under the same compression constraints, `QSLR` improves upon the best sequential baseline, reducing WikiText-2 perplexity from 18.98 to 15.41 on Llama-3.2-1B at 2.51 BPW, and KL divergence to the dense model from 43.79×10^{-3} to 41.94×10^{-3} on Qwen3.8-27B at 3.71 BPW.
2. **Mixed sparse and low-rank allocation** We demonstrate that for many scalar quantization format, using the mixed representation with sparsity and low-rank corrections can preserve model quality better than allocating it entirely to either component alone. On Qwen3-4B-Base, using (sym 2-bit) quantizer and 2.47 BPW, the $\mathbf{Q} + \mathbf{S} + \mathbf{L}$ representation achieves WikiText2 perplxity 9.76 compared to 10.59 for $\mathbf{Q} + \mathbf{S}$ and 10.77 for $\mathbf{Q} + \mathbf{L}$. All three allocations use our ADMM-based solver, isolating the benefits of the representation from the fitting procedure.
3. **GPU kernels and inference acceleration** We develop specialized CUDA-core and tensor-core kernels for the quantized-plus-sparse-plus-low-rank representation using SpQR's double-quantized format. On an NVIDIA RTX 4090 at batch size eight, our kernels achieve end-to-end decoding speedups up to $1.74\times$ over dense FP16 inference for a Llama-30B model using SpQR(3,16,3,16) double quantized format (see Section 2), 1% sparsity, and rank-16 corrections.

2 Related Work

Individual activation-aware solvers and sequential fitting. GPTQ (Frantar et al., 2023) quantizes weights while compensating for their effect on the layer output using calibration activations. SparseGPT (Frantar & Alistarh, 2023) employs a similar error-correction method but targets pruning

instead, while SVD-LLM (Wang et al., 2025) studies activation-aware low-rank compression. Each solver targets one compression primitive: quantization, sparsity, or low-rank compression. A simple way to combine these primitives is to fit them sequentially to the remaining residual. For example, first fit (activation-aware manner) $\mathbf{Q}$ to $\mathbf{W}$, then fit $\mathbf{S}$ to $\mathbf{W} - \mathbf{Q}$, and finally fit $\mathbf{L}$ to $\mathbf{W} - \mathbf{Q} - \mathbf{S}$.

Pairwise fitting solvers. Pairwise methods combine two compression primitives in a single representation. **(i) Quantization and sparse outliers ($\mathbf{Q} + \mathbf{S}$):** SpQR (Dettmers et al., 2023) combines low-bit quantization with high-precision storage for outliers. It partitions weights into groups with their own scales and zero points, then quantizes these parameters in secondary groups as well. We denote this format by $\mathrm{SpQR}(b, g, b_s, g_s)$: weight bit width b, weight group size g, scale/zero-point bit width b_s, and secondary group size g_s. SpQR selects outliers using their estimated contribution to reconstruction error. A hyperparameter τ sets its relative outlier threshold. Thus, τ is an error threshold that can be calibrated to provide an approximate number of high-precision elements. **(ii) Quantization and low rank ($\mathbf{Q} + \mathbf{L}$):** LoftQ (Li et al., 2024), LQ-LoRA (Guo et al., 2024), and CALDERA (Saha et al., 2024) combine quantization with low-rank corrections. The three procedures employ alternating minimization: LoftQ uses a data-free version; LQ-LoRA integrates the model's gradients for the decomposition; and CALDERA targets the layerwise reconstruction error. **(iii) Sparsity and low rank ($\mathbf{S} + \mathbf{L}$):** OATS (Zhang & Papyan, 2025), HASSLE-free (Makni et al., 2025a), and 3BASiL (Makni et al., 2025b) combine sparse and low-rank components. OATS alternates sparse and low-rank fitting under an outlier-aware setting, reporting improvements over pure pruning alone at matched compression levels. HASSLE-free uses alternating minimization and 3BASiL employs ADMM to target the layerwise reconstruction error.

Non-uniform compression allocation. Designing specialized solvers under a *per-layer* budget constraint and choosing how to allocate the compression budget across the layers of a given LLM are separate research directions. Malinovskii et al. (2024) derive a sensitivity-weighted error model from their Linearity Theorem and use it to allocate per-layer quantization precision under a *global* budget constraint. We adopt this allocation framework and expand its candidates to include sparsity and low rank, as described in subsection 4.6.

3 QSLR: A *joint* OPTIMIZATION PROCEDURE FOR A QUANTIZED + SPARSE + LOW-RANK REPRESENTATION

3.1 PROBLEM FORMULATION

We compress a pretrained LLM sequentially layer by layer using calibration sequences, as in GPTQ (Frantar et al., 2023). For a given linear layer $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, we collect its input activations from the preceding compressed layers in $\mathbf{X} \in \mathbb{R}^{N \times d_{\text{in}}}$, where N is the number of calibration tokens. We seek a *joint* quantized-plus-sparse-plus-low-rank decomposition $\widehat{\mathbf{W}} = \mathbf{Q} + \mathbf{S} + \mathbf{L}$ minimizing the *layerwise reconstruction error*

$$\mathcal{L}(\mathbf{Q}, \mathbf{S}, \mathbf{L}) = \frac{1}{2}\left\|\mathbf{X}(\mathbf{W} - \mathbf{Q} - \mathbf{S} - \mathbf{L})\right\|_F^2 \quad \text{s.t. } \mathbf{Q} \in \mathcal{Q}_b, \ \ \|\mathbf{S}\|_0 \le k, \ \ \operatorname{rank}(\mathbf{L}) \le r. \tag{1}$$

Here $\mathcal{Q}_b$ denotes the set of matrices representable by the chosen quantization format at bit-width b. For example, a b-bit symmetric uniform grid uses a fixed scale $\boldsymbol{s}_j > 0$ for column j, with each entry of that column given by $\boldsymbol{s}_j z$ for an integer code $z \in \{-2^{b-1}, \ldots, 2^{b-1}-1\}$. The budget $\|\mathbf{S}\|_0 \le k$ caps the number of non-zero full-precision elements; and $\mathbf{L}$ has rank at most r.

3.2 ALGORITHM DERIVATION

We use the Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011) to jointly optimize $\mathbf{Q}$, $\mathbf{S}$, and $\mathbf{L}$ in Equation (1), following its successful use in layerwise reconstruction problems under combinatorial constraints (Meng et al., 2024; Makni et al., 2025b; Lucas et al., 2026). The main challenge is to handle the quantization, sparsity, and low-rank constraints together. We introduce an auxiliary variable $\mathbf{V}$ as a copy of $\mathbf{Z} := \mathbf{Q} + \mathbf{S}$, with the constraint $\mathbf{V} = \mathbf{Z}$, and keep $\mathbf{L}$ as an additive low-rank correction. The associated augmented Lagrangian is given in Equation (2). We separate the procedure into two stages. Initialization fits a weighted low-rank approximation and constructs the format-specific quantization set $\mathcal{Q}_b$ from its residual. The subsequent ADMM stage keeps this set fixed: scales, zero points, codebooks, etc., while code assignments, sparse support and values, and low-rank factors are updated. Section A details the format-specific initializations.

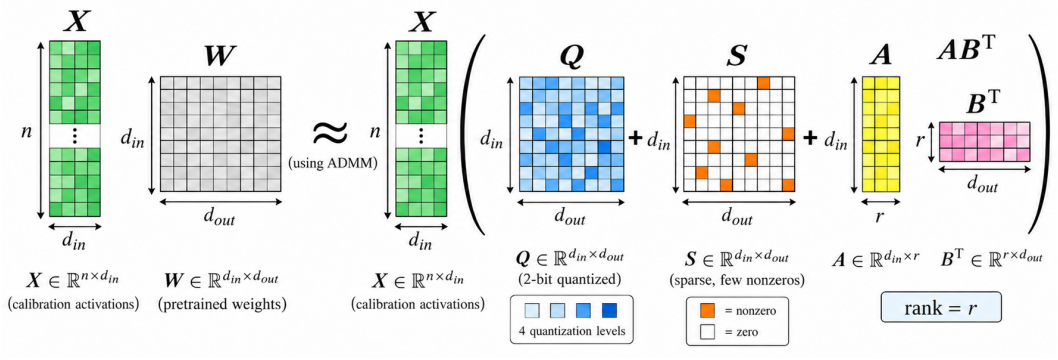

Figure 1: Illustration of the activation-aware decomposition $\mathbf{XW} \approx \mathbf{X}(\mathbf{Q} + \mathbf{S} + \mathbf{L})$ obtained using ADMM, where $\mathbf{Q}$ is quantized, $\mathbf{S}$ is sparse, and $\mathbf{L} = \mathbf{AB}^\top$ is a rank-r term.

Using the damped second-moment matrix $\mathbf{H} = \mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}$, with $\lambda > 0$, we write the augmented Lagrangian with penalty $\rho > 0$ and scaled dual variable $\mathbf{U}$ as

$$\begin{aligned}\mathcal{L}_\rho(\mathbf{V}, \mathbf{Q}, \mathbf{S}, \mathbf{L}, \mathbf{U}) = \frac{1}{2}\left\|\mathbf{H}^{1/2}(\mathbf{W} - \mathbf{V} - \mathbf{L})\right\|_F^2 + \mathbb{I}_{\mathcal{Q}_b}(\mathbf{Q}) + \mathbb{I}_{\Sigma_k}(\mathbf{S}) + \mathbb{I}_{\mathcal{R}_r}(\mathbf{L}) \\ + \frac{\rho}{2}\left\|\mathbf{V} - (\mathbf{Q} + \mathbf{S}) + \mathbf{U}\right\|_F^2 - \frac{\rho}{2}\|\mathbf{U}\|_F^2.\end{aligned} \tag{2}$$

Here $\mathbb{I}_{\mathcal{A}}$ equals zero for a matrix in $\mathcal{A}$ and $+\infty$ otherwise; Σ_k contains matrices with at most k nonzero entries, and $\mathcal{R}_r$ contains matrices of rank at most r. The first term is the reconstruction error substituted with the soft-copy $\mathbf{V}$. The second quadratic penalty enforces agreement between $\mathbf{V}$ and $\mathbf{Q} + \mathbf{S}$. We update $\mathbf{V}$, $(\mathbf{Q}, \mathbf{S})$, and $\mathbf{L}$ cyclically, followed by a dual update, as described below and in Algorithm 1.

(i) $\mathbf{V}$ (smooth consensus). With $\mathbf{Q}, \mathbf{S}, \mathbf{L}$ fixed, the augmented objective is quadratic in $\mathbf{V}$ and its gradient vanishes at the single linear solve $(\mathbf{H} + \rho\mathbf{I})\mathbf{V} = \mathbf{H}(\mathbf{W} - \mathbf{L}) + \rho(\mathbf{Q} + \mathbf{S} - \mathbf{U})$. We use an efficiency idea introduced by Meng et al. (2024) and eigendecompose $\mathbf{H} = \mathbf{EDE}^\top$ once and reuse it: $(\mathbf{H} + \rho\mathbf{I})^{-1} = \mathbf{E}(\mathbf{D} + \rho\mathbf{I})^{-1}\mathbf{E}^\top$, so changing ρ only requires inverting a diagonal matrix.

(ii) $(\mathbf{Q}, \mathbf{S})$ (joint quantize + sparsify). We update $(\mathbf{Q}, \mathbf{S})$ as one block, with feasible sums in

$$\mathcal{C} := \mathcal{Q}_b + \Sigma_k = \left\{\mathbf{Q} + \mathbf{S} : \mathbf{Q} \in \mathcal{Q}_b,\ \|\mathbf{S}\|_0 \le k\right\}, \tag{3}$$

With $\mathbf{V}$ and $\mathbf{U}$ fixed, this update seeks the closest matrix $\mathbf{Q} + \mathbf{S} \in \mathcal{C}$ to $\mathbf{P} = \mathbf{V} + \mathbf{U}$. Our update first quantizes this target and then corrects its k largest-magnitude residual entries. We denote this operation by $\text{QSPROX}(\mathbf{P}; \mathcal{Q}_b, k)$:

$$\mathbf{Q} = \text{Quant}_{\mathcal{Q}_b}(\mathbf{P}), \qquad \mathbf{S} = \text{TopK}_k(\mathbf{P} - \mathbf{Q}). \tag{4}$$

Here $\text{Quant}_{\mathcal{Q}_b}$ is the quantizer associated with the fixed format, and TopK_k retains the k largest entries in absolute value across the entire matrix and sets the rest to zero.

When $\mathcal{Q}_b$ is a *fixed Cartesian product of scalar grids* (i.e., each entry is assigned independently to a fixed and finite set of values), the quantizer simply rounds each entry to its nearest grid value. In this case, quantization followed by top-k residual correction is an *exact joint projection*, as shown below. This includes per-channel and groupwise uniform quantization, and SpQR (Dettmers et al., 2023) double-quantized format. It additionally includes various other formats like NormalFloat format (Dettmers et al., 2024) or NVFP4 (Alvarez et al., 2025). Each weight is independently assigned to the nearest value from a fixed scalar grid.

Proposition 3.1 (QSPROX returns an exact Euclidean projection onto $\mathcal{C}$)**.** *Suppose $\mathcal{Q}_b$ is a Cartesian product of fixed scalar grids and the quantizer rounds each entry to a nearest grid value. For any $\mathbf{P}$, $\text{QSPROX}(\mathbf{P}; \mathcal{Q}_b, k)$ returns a minimizer of $\|\mathbf{P} - \mathbf{Q} - \mathbf{S}\|_F^2$ over $\mathbf{Q} \in \mathcal{Q}_b$, $\|\mathbf{S}\|_0 \le k$. The projection is in general* set-valued*, and QSPROX returns one such minimizer.*

Proof. Fix a support T, $|T| \le k$, and minimize over $(\mathbf{Q}, \mathbf{S})$ with $\text{supp}\,\mathbf{S} \subseteq T$; the objective separates over entries. Off T, $\mathbf{S}_{ij} = 0$, so entry (i, j) gives $\min_{q \in \text{grid}_{ij}}(\mathbf{P}_{ij} - q)^2 = (\mathbf{P}_{ij} - q_{ij})^2$,

where q_{ij} is a nearest point in the fixed scalar grid for entry (i, j). On T, $\mathbf{S}_{ij}$ is unconstrained, so $\mathbf{S}_{ij} = \mathbf{P}_{ij} - \mathbf{Q}_{ij}$ zeros that entry's contribution for any grid value. Hence, for fixed T, the minimum is $\sum_{(i,j)\notin T}(\mathbf{P}_{ij} - q_{ij})^2$, (round-to-nearest applied *off the support*). Minimizing over $|T| \le k$ removes the k largest entries of $\mathbf{P} - q$ in absolute value, where $q = \mathrm{RTN}(\mathbf{P})$, which is exactly what QSPROX does. Hence, $\mathbf{Q} + \mathbf{S}$ is a nearest point of $\mathcal{C}$. □

(iii) L (low-rank, C_F-gated). With $\mathbf{V}$ fixed, the low-rank subproblem minimizes $\|\mathbf{H}^{1/2}(\mathbf{W} - \mathbf{V} - \mathbf{L})\|_F^2$ subject to $\mathrm{rank}\,(\mathbf{L}) \le r$. This subproblem has a closed-form solution (see discussion in 3BASiL (Makni et al., 2025b)) through weighted singular value decomposition, denoted by `wSVD` (activation-aware low-rank correction). For a residual matrix $\mathbf{R}$, we define

$$\texttt{wSVD}_r(\mathbf{R}, \mathbf{H}) := \mathbf{H}^{-1/2} P_r\Big(\mathbf{H}^{1/2}\mathbf{R}\Big) \in \underset{\mathrm{rank}(\mathbf{M})\le r}{\arg\min} \big\|\mathbf{H}^{1/2}(\mathbf{R} - \mathbf{M})\big\|_F^2, \tag{5}$$

where P_r denotes a best rank-r approximation in Frobenius norm, obtained by retaining the leading r singular components. The same cached eigendecomposition of $\mathbf{H}$ can be reused for every `wSVD` call since we have $\mathbf{H}^{\pm 1/2} = \mathbf{E}\mathbf{D}^{\pm 1/2}\mathbf{E}^\top$. The candidate update is therefore $\mathbf{L}^\star = \texttt{wSVD}_r(\mathbf{W} - \mathbf{V}, \mathbf{H})$. In practice, we approximate P_r using randomized SVD (Halko et al., 2011) for numerical efficiency similar to LQ-LoRA (Guo et al., 2024) and 3BASiL (Makni et al., 2025b). We commit the candidate only when its move is bounded by the consensus move.

$$(C_F\text{-gate}) \quad \text{accept } \mathbf{L} \leftarrow \mathbf{L}^\star \text{ iff } \|\mathbf{L}^\star - \mathbf{L}\|_F \le C_F \|\mathbf{V} - \mathbf{V}_{\text{prev}}\|_F, \text{ else keep } \mathbf{L}. \tag{6}$$

(iv) Scaled dual update. The scaled dual update is $\mathbf{U} \leftarrow \mathbf{U} + \mathbf{V} - (\mathbf{Q} + \mathbf{S})$. Whenever the penalty increases from ρ to $\eta\rho$, we also rescale $\mathbf{U} \leftarrow \mathbf{U}/\eta$ to preserve the Lagrange multiplier $\rho\mathbf{U}$.

Convergence Guarantee. The key design choice is to make $\mathbf{V}$ a smooth copy of the entire $\mathbf{Q} + \mathbf{S}$ block. This preserves the smooth projection and low-rank structure as additive to the smooth part of 3BASiL (Makni et al., 2025b), so its convergence argument applies with the sparse projection replaced by a joint projection onto $\mathcal{C}$ (quantized-plus-sparse set). For fixed scalar grids, QSPROX computes this projection exactly (Proposition 3.1). This gives the following guarantee; Section B identifies the required changes to adapt the analysis.

Theorem 3.2 (Convergence of Algorithm 1 under a fixed scalar grid)**.** *Let $\mathbf{H} \succ 0$ and let $\mathcal{Q}_b$ be a Cartesian product of fixed finite scalar grids. Consider the iterates of Algorithm 1 with $\rho_0 > 0$ multiplied by $\eta > 1$ every fixed $\kappa \ge 1$ iterations. Then $\mathbf{Z}^{(t)} := \mathbf{Q}^{(t)} + \mathbf{S}^{(t)}$ and $\mathbf{V}^{(t)}$ converge to a common limit $\bar{\mathbf{Z}} \in \mathcal{C}$, and $\mathbf{L}^{(t)} \to \bar{\mathbf{L}}$ with $\mathrm{rank}\,(\bar{\mathbf{L}}) \le r$. Thus, $\mathbf{Z}^{(t)} + \mathbf{L}^{(t)}$ converges to a feasible decomposition. Moreover, for some constant $K < \infty$ and all sufficiently large t,*

$$\max_{\mathbf{A}\in\{\mathbf{V},\mathbf{Z},\mathbf{L}\}} \|\mathbf{A}^{(t+1)} - \mathbf{A}^{(t)}\|_F \le K/\rho_{t-1}, \quad \|\mathbf{V}^{(t+1)} - \mathbf{Z}^{(t+1)}\|_F \le K/\rho_{t-1}.$$

Both bounds are $O(\eta^{-\lfloor t/\kappa \rfloor})$ using Algorithm 1 chosen penalty scheduler.

4 Numerical Experiments

4.1 Notation and Baselines

Allocation details. We report bits per weight (BPW) over the compressed attention and MLP weight matrices, excluding other LLM weights (e.g., embeddings and the output head). Sparse values and low-rank factors use 16-bit storage. We follow SpQR's reference BPW calculator and use the Compressed Sparse Row (CSR) format for sparse corrections.[1] For rank r, the storage is

$$B = B_{\mathcal{Q}} + 32k + 16r(m + n), \tag{7}$$

where $B_{\mathcal{Q}}$ includes quantized weights and format metadata. Thus, increasing r by one trades approximately $(m + n)/2$ sparse entries at a fixed BPW. We specify configurations as $(\mathbf{Q}, \mathrm{sp}, \mathrm{rk})$.

Baseline definitions. The first baseline we introduce is `Best_seq` which applies the activation-aware solvers GPTQ (quantization), SparseGPT (pruning), and `wSVD` (low-rank) to the running residual. We evaluate all six fitting orders, use the same order for all model layers, and report the

[1] SpQR BPW calculator: `get_average_number_of_bits`, `main.py`, lines 35–69.

Algorithm 1 QSLR: initialization followed by fixed-grid ADMM for one linear layer. Section A gives the format-specific initialization of the $\mathcal{Q}_b$ grid.

Input: pretrained weight $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$; damped second-moment matrix $\mathbf{H} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$; **Compression Parameters:** bit-width b; sparsity budget k; rank r; **ADMM params.:** init penalty $\rho > 0$ with multiplier $\eta \geq 1$ updated every κ iters; low-rank gate C_F; stop tol. ϵ; max iters T.
Output: $\widehat{\mathbf{W}} = \mathbf{Q} + \mathbf{S} + \mathbf{L}$ with $\mathbf{Q}$ in $\mathcal{Q}_b$, $\|\mathbf{S}\|_0 \leq k$, $\operatorname{rank}(\mathbf{L}) \leq r$.

$\mathbf{H} = \mathbf{E}\mathbf{D}\mathbf{E}^\top$ // decompose once; reuse in consensus and every wSVD call
Stage 1: Initialize the decomposition & freeze $\mathbf{Q}_b$
  $\mathbf{L} \leftarrow \texttt{wSVD}_r(\mathbf{W}, \mathbf{H})$; $\mathbf{R} \leftarrow \mathbf{W} - \mathbf{L}$
  $\mathcal{Q}_b \leftarrow \text{InitSet}(\mathbf{R}, \mathbf{H}; \text{format}, b, k)$ // format-specific; see subsection A.1
  $(\mathbf{Q}, \mathbf{S}) \leftarrow \text{QSPROX}(\mathbf{R}; \mathcal{Q}_b, k)$; $\mathbf{V} \leftarrow \mathbf{Q} + \mathbf{S}$; $\mathbf{V}_{\text{prev}} \leftarrow \mathbf{V}$; $\mathbf{U} \leftarrow \mathbf{0}$; $\widehat{\mathbf{W}} \leftarrow \mathbf{Q} + \mathbf{S} + \mathbf{L}$

Stage 2: Optimize on the fixed quantization set $\mathbf{Q}_b$
for $t = 1, \ldots, T$ **do**
    $\mathbf{V}_{\text{prev}} \leftarrow \mathbf{V}$
    $\mathbf{V} \leftarrow \mathbf{E}(\mathbf{D} + \rho\mathbf{I})^{-1}\mathbf{E}^\top\big(\mathbf{H}(\mathbf{W} - \mathbf{L}) + \rho\,(\mathbf{Q} + \mathbf{S} - \mathbf{U})\big)$ // (i) diagonal inversion
    $(\mathbf{Q}, \mathbf{S}) \leftarrow \text{QSPROX}\big(\mathbf{V} + \mathbf{U}; \mathcal{Q}_b, k\big)$ // (ii) quantize-then-sparsify residual
    $\mathbf{L}^\star \leftarrow \texttt{wSVD}_r(\mathbf{W} - \mathbf{V}, \mathbf{H})$ // (iii) low-rank update
    if $\|\mathbf{L}^\star - \mathbf{L}\|_F \leq C_F\,\|\mathbf{V} - \mathbf{V}_{\text{prev}}\|_F$ **then** $\mathbf{L} \leftarrow \mathbf{L}^\star$ // commit only on bounded move
    $\mathbf{U} \leftarrow \mathbf{U} + \mathbf{V} - (\mathbf{Q} + \mathbf{S})$ // (iv) scaled dual update
    every κ **iters:** $\rho \leftarrow \eta\rho$; $\mathbf{U} \leftarrow \mathbf{U}/\eta$ // penalty update and dual rescaling
    $\widehat{\mathbf{W}}_{\text{prev}} \leftarrow \widehat{\mathbf{W}}$; $\widehat{\mathbf{W}} \leftarrow \mathbf{Q} + \mathbf{S} + \mathbf{L}$; **stop if** $\|\widehat{\mathbf{W}} - \widehat{\mathbf{W}}_{\text{prev}}\|_F < \epsilon\,\|\widehat{\mathbf{W}}_{\text{prev}}\|_F$

return $\widehat{\mathbf{W}} = \mathbf{Q} + \mathbf{S} + \mathbf{L}$

lowest WT2 perplexity over the six runs. We denote our rank-zero solver for $\mathbf{Q} + \mathbf{S}$ by QS_ADMM and define two additional baselines. QS_ADMM+LR first fits $\mathbf{Q} + \mathbf{S}$, freezes both components, and then adds a rank-r wSVD correction to $\mathbf{W} - \mathbf{Q} - \mathbf{S}$. LR+QS_ADMM first fits $\mathbf{L} = \texttt{wSVD}_r(\mathbf{W}, \mathbf{H})$, freezes it, and then fits $\mathbf{Q} + \mathbf{S}$ to $\mathbf{W} - \mathbf{L}$ using the rank-zero solver. Neither order revisits the components fitted in the first stage. We introduce these two baselines to isolate the gains from the *joint* optimization provided by QSLR: they use the same ADMM solver for $\mathbf{Q}+\mathbf{S}$ and the same wSVD operation for $\mathbf{L}$, but fit them in separate stages rather than allowing all three components to adapt together. Subsection C.1 gives both definitions and the residual-fitting pseudocode for Best_seq.

Similarly, SpQR+LR first fits SpQR and then adds a wSVD correction to its residual. LR+SpQR reverses these steps: it fits the low-rank component first, then applies SpQR to the residual.

4.2 Setup and Evaluation

We organize the experiments into four subsections. In subsection 4.3, we establish the effectiveness of our joint optimization approach under a symmetric uniform grid using QSLR to fit a given $\mathbf{Q} + \mathbf{S} + \mathbf{L}$ configuration. We show that our proposed method largely outperforms sequentially using mature solvers from quantization, pruning, and low-rank compression. For Qwen3-4B-Base with ($\mathbf{Q}$=sym W2, sp=1%, rk=16) at 2.47 BPW, QSLR achieves WT2 perplexity 9.77 versus 13.77 for the Best_seq baseline (Table 1). In subsection 4.4, we test whether the benefits of joint fitting extend to SpQR's quantization format, comparing QSLR against SpQR at rank zero and against SpQR+LR and LR+SpQR when low-rank corrections are included. For Qwen3.8-27B using SpQR$(2, 16, 3, 16)$ and rank 8, QSLR reaches WT2 perplexity 7.77 at 2.98 BPW, compared with 7.85 for SpQR+LR at the same BPW and 7.05 for dense FP16 (Table 2). Its WT2 KL is 149.36×10^{-3}, compared with 160.43×10^{-3} for SpQR+LR. In subsection 4.5, we ask whether a fixed correction budget is better allocated to sparsity, low rank, or a combination of both. On Llama-3.2-1B with (sym W2) quantization at 2.51 BPW, $\mathbf{Q} + \mathbf{S} + \mathbf{L}$ achieves WT2 perplexity 15.20, compared with 20.01 for $\mathbf{Q} + \mathbf{S}$ and 17.92 for $\mathbf{Q} + \mathbf{L}$. We use QSLR for all allocations to isolating the benefit of the mixed representation from improvements in the fitting procedure.

We complement these comparisons with two experimental studies deferred to the Appendix. In subsection C.4, we extend QSLR to the E8 lattice codebook and compare against CALDERA ($\mathbf{Q}+\mathbf{L}$ specialized solver). On Llama-3.2-1B at 2.53 BPW, our $\mathbf{Q}+\mathbf{L}$ solver achieves WT2 perplexity 14.35

versus 16.34 for CALDERA. In subsection C.5, we study data-free NormalFloat2 (Dettmers et al., 2024) (NF2) initialization followed by 100-step LoRA on Meta-Llama-3-8B. At rank 16 and the same 2.60 BPW, our $\mathbf{Q} + \mathbf{L}$ initialization achieves post-LoRA WT2 perplexity 14.70 versus 20.79 for LoftQ. Adding 1% sparsity improves our mean eight-task accuracy from 49.46% to 54.27%.

Finally, in subsection 4.6, we investigate how much compression can benefit from varying quantization precision, sparsity, and rank across weight matrices under a *global* storage budget. On Llama-3.2-1B, non-uniform allocation achieves C4 ppl 20.42 at 3.00 BPW, compared with 25.58 for uniform W3 at 3.01 BPW and 23.46 for uniform W2 with 1% sparsity and rank 64 at 3.08 BPW.

In all of our experiments, we use 128 WikiText-2 training segments of length 2048 for calibration, with Hessian damping 0.1. `QSLR` uses at most $T = 300$ iterations, initial ADMM penalty set to $\rho = 0.05$ and multiplied by $\eta = 1.05$ every $\kappa = 3$ iterations, and low-rank gate $C_F = 0.3$. Bold values indicate the best performance among compressed methods in each comparison.

4.3 QSLR WITH A UNIFORM SCALAR QUANTIZER

We first establish the effectiveness of our proposed joint fitting procedure using a symmetric per-channel quantizer. We compare with `Best_seq`, `QS_ADMM+LR`, and `LR+QS_ADMM`, defined above, under the same quantization, sparsity, and rank constraints. This tests whether fitting all components jointly improves on sequentially using separate solvers for $\mathbf{Q}$, $\mathbf{S}$ and $\mathbf{L}$ in any order.

`QSLR` gives the lowest perplexity in every W2 setting of Table 1. For Llama-3.2-1B with ($\mathbf{Q}$=W2, sp=1%, rk=16), WT2 PPL is reduced from 41.61 for `Best_seq`, 18.98 for `QS_ADMM+LR`, and 21.25 for `LR+QS_ADMM` to 15.41 with `QSLR`. Under this configuration, `QSLR` reduces the WT2 perplexity gap to the dense model by 82.22% compared with `Best_seq`.

Table 1: Llama-3.2-1B and Qwen3-4B-Base: joint versus sequential fitting with ($\mathbf{Q}$=Wb, sp=c%, rk=r). WT2 perplexity (↓). `QS` abbreviates `QS_ADMM`.

Config			Llama-3.2-1B					Qwen3-4B-Base				
Q	sp	rk	BPW	`Best_seq`	`QS+LR`	`LR+QS`	`QSLR`	BPW	`Best_seq`	`QS+LR`	`LR+QS`	`QSLR`
W2	1%	16	2.51	41.61	18.98	21.25	**15.41**	2.47	13.77	10.63	10.93	**9.77**
W2	1%	32	2.70	36.27	18.43	19.44	**14.56**	2.62	13.52	10.56	10.83	**9.53**
W2	5%	16	3.79	27.47	13.81	13.83	**12.80**	3.75	11.37	9.19	9.24	**8.90**
W2	5%	32	3.98	26.09	13.77	13.62	**12.50**	3.90	11.34	9.16	9.17	**8.82**
Dense FP16			PPL = 9.75					PPL = 7.90				

4.4 QSLR IN THE SPQR FORMAT

Having established the benefits of joint fitting using a uniform quantizer, we now test whether `QSLR` gains extend to a specialized pairwise solver. To this end, we use SpQR with format $(b, 16, 3, 16)$, which fits $\mathbf{Q} + \mathbf{S}$ simultaneously (Section 2). SpQR selects outliers whose estimated quantization error exceeds a threshold controlled by τ, so the number of sparse entries is not specified in advance (see Section 2). For a fair comparison, we fix $\tau = 1.0$ and retain SpQR's measured number of outliers in each weight matrix as the sparse budget for `QSLR`. We introduce the two additional baselines `SpQR+LR` and `LR+SpQR` as discussed in the baseline definitions. `QSLR` achieves the lowest WT2 KL divergence in every completed model-rank comparison in Table 2.

On Qwen3-4B-Base at SpQR$(2, 16, 3, 16)$ and rank 16, `QSLR` reduces WT2 KL from 289.39×10^{-3} for the best sequential baseline `SpQR+LR`, to 230.04×10^{-3} yielding a 20.51% reduction.

Additional results are reported in subsection C.2.1.

Table 2: KL(dense‖compressed) $\times 10^3$ (↓).

Rank	Method	Qwen3 4B-Base	Qwen3.8 27B
	W2		
0	`SpQR`	292.33	162.25
	`QSLR`	**259.20**	**157.30**
8	`LR+SpQR`	298.50	159.93
	`SpQR+LR`	294.65	160.43
	`QSLR`	**247.20**	**149.36**
16	`LR+SpQR`	293.09	158.54
	`SpQR+LR`	289.39	160.69
	`QSLR`	**230.04**	**152.86**
	W3		
0	`SpQR`	69.35	46.09
	`QSLR`	**65.77**	**43.14**
8	`LR+SpQR`	62.35	44.84
	`SpQR+LR`	65.92	44.37
	`QSLR`	**56.98**	**41.99**
16	`LR+SpQR`	62.09	43.79
	`SpQR+LR`	64.46	44.51
	`QSLR`	**54.97**	**41.94**

4.5 SPARSITY VS. LOW-RANK TRADEOFFS

In the previous subsections, we focused our experiments to showcase the QSLR improved fitting. For a pure $(\mathbf{Q}+\mathbf{S})$ compression, QSLR can outperform SpQR given the same constraints, see Table 2. In this section, we fix the total BPW and study how to divide the available budget between $\mathbf{S}$ and $\mathbf{L}$ using QSLR. At the endpoints, all correction storage goes to sparsity $(\mathbf{Q}+\mathbf{S})$ or to low rank $(\mathbf{Q}+\mathbf{L})$; and intermediate allocations use both. This isolates the benefits from mixed representation from using an improved solver.

Figure 2 shows symmetric W2 on Llama-3.2-1B and Qwen3-4B-Base at 2.51 and 2.47 BPW, and SpQR $(2, 16, 3, 16)$ on Llama-3.2-1B at 2.96 BPW. The symmetric budgets are anchored by 1% sparsity and rank 16; the SpQR budget is set by its threshold-selected sparse allocation.

Using both corrections improves on either endpoint in all three comparisons. For example, on Qwen3-4B-Base at 2.47 BPW, the rank-16 mixed allocation reaches WT2 perplexity 9.76, compared with 10.59 for $\mathbf{Q}+\mathbf{S}$ and 10.77 for $\mathbf{Q}+\mathbf{L}$.

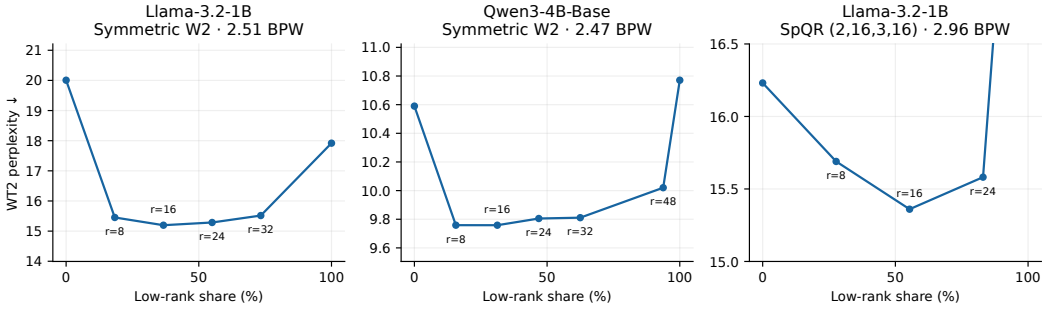

Figure 2: **Sparsity vs. low rank tradeoffs at fixed BPW.** The quantizers used are sym W2 for the left and middle pandel and SpQR $(2, 16, 3, 16)$ for th eright panel. The mixed-precision compression outperforms the endpoints corresponding to $\mathbf{Q}+\mathbf{S}$ and $\mathbf{Q}+\mathbf{L}$ in these settings.

4.6 INTEGRATION WITH NON-UNIFORM ALLOCATION

The previous sections have shown the benefits of the additional flexibility provided by using the mixed representation and fitting using QSLR. The preceding allocation comparisons use uniform configurations across LLM weight matrices. We dedicate this section to better understand the benefits of allocating different *per-layer* budgets for all three components simultaneously under a given *global* compression budget. To this end, we build upon the non-uniform quantization allocation framework of Malinovskii et al. (2024) and extend it to our $\mathbf{Q}+\mathbf{S}+\mathbf{L}$ representation. Under the assumptions of their Linearity Theorem, the increase in perplexity is approximated by a sum of per-matrix squared reconstruction errors weighted by sensitivity coefficients. In their quantization-only formulation, the choice for matrix $\mathbf{W}_\ell$ is its bit width b_ℓ. Our adaptation extends the candidate choices to a configuration $c = (b, k, r)$, specifying the bit width of $\mathbf{Q}$, sparse-entry count of $\mathbf{S}$, and rank of $\mathbf{L}$.

For each matrix and candidate configuration, we first fit using QSLR and obtain $\widehat{\mathbf{W}}_{\ell,c} = \mathbf{Q}_{\ell,c} + \mathbf{S}_{\ell,c} + \mathbf{L}_{\ell,c}$. Let $B_{\ell,c}$ be its storage in bits, including format metadata and sparse indices, and let $e_{\ell,c} = \|\mathbf{W}_\ell - \widehat{\mathbf{W}}_{\ell,c}\|_F^2 / \|\mathbf{W}_\ell\|_F^2$. Following their discrete allocation formulation (Malinovskii et al., 2024), we select one candidate per matrix by solving

$$\min_{c_\ell \in \mathcal{C}_\ell} \sum_\ell \alpha_\ell e_{\ell,c_\ell} \quad \text{subject to} \quad \sum_\ell B_{\ell,c_\ell} \le \beta \sum_\ell |\mathbf{W}_\ell|, \tag{8}$$

where β is the target BPW and $|\mathbf{W}_\ell|$ is the number of weights in matrix ℓ. We estimate each sensitivity coefficient α_ℓ by fitting the perplexity increase against the relative squared magnitude of random weight perturbations exactly similar to Malinovskii et al. (2024). We employ the same integer allocation solver used by the authors which relies on the saved candidate errors and storage costs. Restricting every candidate to $k = r = 0$ recovers their quantization-only allocation formulation.

We then sequentially refit the selected configurations using QSLR, with activations from the compressed preceding layers. On Llama-3.2-1B, the allocation in Figure 3 selects from $\mathbf{Q} \in \{\text{sym W2}, \text{W3}, \text{W4}\}$, $\text{sp} \in \{0, 0.1, 0.5, 1\}\%$, and $\text{rk} \in \{0, 8, 16, 64\}$. At 3.00 BPW, the non-uniform allocation yields C4 perplexity 20.42, compared with 25.58 for uniform W3 at 3.01 BPW and 23.46 for the uniform $(\mathbf{Q}{=}\text{W2}, \text{sp}{=}1\%, \text{rk}{=}64)$ configuration at 3.08 BPW. At the lowest plotted budget, however, non-uniform allocation reaches 29.67 at 2.50 BPW, while uniform $(\mathbf{Q}{=}\text{W2}, \text{sp}{=}1\%, \text{rk}{=}16)$ reaches 28.62 at 2.52 BPW. Interestingly, this is consistent with the limitation discussed by Malinovskii et al. (2024, Section 6.1): at low bit widths, larger compression errors can invalidate the small-error approximation underlying the allocation model. That said, the gains at the valid range (> 2.60 BPW for Llama-3.2-1B) yield important utility gains as discussed above (see Figure 3).

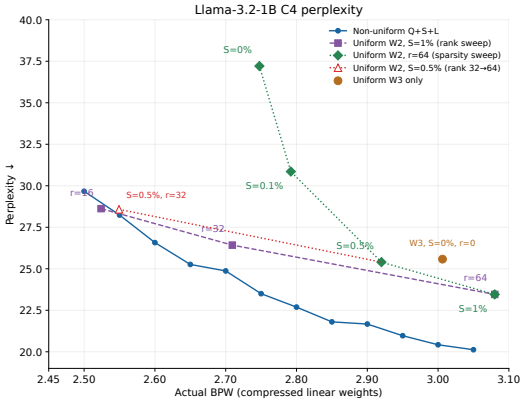

Figure 3: **Non-uniform allocation improves the BPW–utility tradeoff.**

4.7 GPU KERNELS AND DECODING SPEED.

We implement two GPU kernels for our introduced decomposition, where $\mathbf{Q} + \mathbf{S}$ follows SpQR's double-quantized format: a CUDA-core implementation and a tensor-core implementation. Their relative performance depends on matrix shape, motivating an automatic dispatch policy that selects between them for each shape. We benchmark $(\mathbf{Q}{=}\text{SpQR}(3, 16, 3, 16), \text{sp}{=}1\%, \text{rk}{=}16)$ with batch size 8 on one NVIDIA RTX 4090. Speedup is FP16 dense latency divided by compressed latency.

Table 3: **GPU benchmark speedups over FP16 dense.** We consider the following decomposition $(\mathbf{Q}{=}\text{SpQR}(3, 16, 3, 16), \text{sp}{=}1\%, \text{rk}{=}16)$ format, and show the speedups on an RTX 4090 GPU obtained using our CUDA cores and Tensor cores implementation **(left)** for matrix-multiplications with different matrix shapes and **(right)** for the end-to-end decoding speed of one token for Llama 7B, 30B, and 70B and batch size set to 8. To make sure the models fit on a single GPU, we use only 4 transformer blocks.

Matrix shape	CUDA	Tensor
4096×4096	$0.69\times$	$0.56\times$
4096×11008	$2.12\times$	$2.05\times$
11008×4096	$1.94\times$	$2.07\times$
16384×16384	$2.31\times$	$2.55\times$

Model	CUDA	Tensor	Auto
Llama 7B	$1.40\times$	$1.33\times$	$1.42\times$
Llama 30B	$1.50\times$	$1.74\times$	$1.74\times$
Llama 70B	$1.53\times$	$1.68\times$	$1.68\times$

5 CONCLUSION AND LIMITATIONS

We introduce QSLR for joint quantized, sparse, and low-rank decomposition. Across multiple experiments, we show both the benefit of joint fitting using QSLR as well as the benefit of the mixed representation compared to a pure $\mathbf{Q}{+}\mathbf{S}$ or a $\mathbf{Q}{+}\mathbf{L}$ compression. We study the effect of non-uniform allocation of quantization, sparsity, and low-rank across layers which provides further utility gains. Finally, we design specialized GPU kernels with decoding speedups for the mixed representation.

Limitations. The main limitations of our proposed algorithm that the used quantized-plus-sparse projection is exact for scalar quantization grids but that does not cover vector quantization grids; we leave designing efficient projection algorithms onto the quantized-plus-sparse set for vectoru quantization grids for future work. Furthermore, we have observed that non-uniform allocation can become inaccurate at very low budgets. Future research direction can look into dedicated methods that target this extremely aggressive regime and that are flexible enough to be adapted to our mixed-representation setting.

AI USE STATEMENT

Generative AI assisted with code development, manuscript revision and polishing, and figure plotting and formatting. The authors are responsible for the final content and verification of the reported results.

REPRODUCIBILITY STATEMENT

The experimental section and appendix specify model families, quantization formats, calibration data, hyperparameters, and evaluation protocols. We will release the code of our algorithm and experimental setups if the paper is accepted.

REFERENCES

Eduardo Alvarez, Omri Almog, Eric Chung, Simon Layton, Dusan Stosic, Ronny Krashinsky, and Kyle Aubrey. Introducing NVFP4 for efficient and accurate low-precision inference. NVIDIA Technical Blog, June 2025. URL https://developer.nvidia.com/blog/?p=102000.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

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. *arXiv preprint arXiv:2306.03078*, 2023.

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

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

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.

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

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53 (2):217–288, 2011.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LzPWWPAdY4.

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

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: Training-free compensation for compressed LLM with eigenspace low-rank approximation, 2025. URL https://arxiv.org/html/2410.21271v3.

Ryan Lucas, Mehdi Makni, Xiang Meng, Adam Deng, and Rahul Mazumder. ADMM-q: An improved hessian-based weight quantizer for post-training quantization of large language models. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=pXd1ueLloD.

Mehdi Makni, Kayhan Behdin, Zheng Xu, Natalia Ponomareva, and Rahul Mazumder. A unified framework for sparse plus low-rank matrix decomposition for LLMs. In *The Second Conference on Parsimony and Learning (Proceedings Track)*, 2025a. URL https://openreview.net/forum?id=hyN75SAJTI.

Mehdi Makni, Xiang Meng, and Rahul Mazumder. 3BASil: An algorithmic framework for sparse plus low-rank compression of LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=byNNv5Et10.

Vladimir Malinovskii, Andrei Panferov, Ivan Ilin, Han Guo, Peter Richtárik, and Dan Alistarh. Pushing the limits of large language model quantization via the linearity theorem. *arXiv preprint arXiv:2411.17525*, 2024.

Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. Alps: Improved optimization for highly sparse one-shot pruning for large language models. In *Annual Conference on Neural Information Processing Systems, NeurIPS*, 2024.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=lkx3OpcqSZ.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Guiyao Tie, Jiawen Shi, Dingjie Song, Yixiao Huang, Ziji Sheng, Xueyang Zhou, Daizong Liu, Pan Zhou, Yongchao Chen, Ran Xu, et al. Autoresearch ai: Towards ai-powered research automation for scientific discovery. *arXiv preprint arXiv:2605.23204*, 2026.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *International Conference on Machine Learning*, 2024.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=LNYIUouhdt.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094. Curran Associates, Inc., 2024. doi:

10.52202/079017-1650. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5d413e48f84dc61244b6be550f1cd8f5-Paper-Datasets_and_Benchmarks_Track.pdf.

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

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, 2025. URL https://openreview.net/forum?id=HyPofygOCT.

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

A INITIALIZATION AND EXPERIMENTAL DETAILS

Our algorithm has two stages: **(i) Initialization:** We proceed by fitting the low-rank component first, then we construct the quantization set and initialize $\mathbf{S}$ and $\mathbf{Q}$ depending on the quantization grid settings; details are presented in what follows. After this step, the quantization grid is fixed for all variations of the algorithm, and the second stage is shared across all algorithms. **(ii) ADMM Updates:** We follow the common update sequence in Algorithm 1: update the smooth consensus variable, use the quantizer to project $\mathbf{Q}$ onto the fixed quantization grid and update $\mathbf{S}$ using the TopK operation (notice that the sparse mask is dynamic and changes through ADMM iterations), update $\mathbf{L}$ subject to the movement gate, and perform dual ascent. We detail the initialization choices below.

A.1 STAGE 1: INITIALIZATION

The initializer first fits the low-rank component using `wSVD` and forms its residual:

$$\mathbf{L}_0 \leftarrow \texttt{wSVD}_r(\mathbf{W}, \mathbf{H}), \qquad \mathbf{R}_0 = \mathbf{W} - \mathbf{L}_0. \tag{9}$$

For $r = 0$, we set $\mathbf{L}_0 = \mathbf{0}$. We then construct the quantization set from $\mathbf{R}_0$ according to the chosen format.

Uniform quantization. We provisionally mask the k entries with largest $H_{ii}(R_0)_{ij}^2$ and fit one quantization scale per output channel on the remaining residual using MSE clipping, the same procedure used in GPTQ (Frantar et al., 2023). We initialize $\mathbf{Q}_0$ by rounding $\mathbf{R}_0$ to this grid, and $\mathbf{S}_0$ by retaining the k residual entries with largest residuals.

SpQR double-quantized format. We apply the default SpQR grid initializer to $\mathbf{R}_0$ to obtain grouped scales and zero points, including their secondary quantization formats. We then initialize $\mathbf{Q}_0$ and $\mathbf{S}_0$ by quantizing the residual on this grid and applying the same weighted top-k correction. SpQR's double quantized scales and zero points remain fixed throughout ADMM.

E8 lattice quantization. For E8, we apply the initialization in the fixed rotated coordinates described in subsection C.4. After forming the residual $\mathbf{R}_0$ in these coordinates, we fit the codebook scale to the remaining residual using the QuIP# and CALDERA default setting, then quantize this residual to initialize $\mathbf{Q}_0$. When we allow for a sparse budget of k nonzeros, $\mathbf{S}_0$ stores the k largest-magnitude residuals post $\mathbf{Q}_0$ quantization. The transforms and scaled codebook remain fixed during ADMM iterations.

Consensus and dual initialization. In the coordinates used by Algorithm 1, we set $\mathbf{V}_0 = \mathbf{Q}_0 + \mathbf{S}_0$ and $\mathbf{U}_0 = \mathbf{0}$. For pure $\mathbf{Q} + \mathbf{L}$, we set $k = 0$ and $\mathbf{S}_0 = \mathbf{0}$; for pure $\mathbf{Q} + \mathbf{S}$, we set $r = 0$ and $\mathbf{L}_0 = \mathbf{0}$.

B CONVERGENCE WITH A FIXED SCALAR QUANTIZATION SET

The proof of Theorem 3.2 follows the layerwise analysis of 3BASiL (Makni et al., 2025b, Appendix A). The smooth block, projected block, and unscaled dual used in 3BASiL correspond here to $\mathbf{V}_t$, $\mathbf{Z}_t = \mathbf{Q}_t + \mathbf{S}_t$, and $\rho_t \mathbf{U}_t$, respectively. $\mathbf{U}_t$ is the scaled dual after any penalty rescaling at the end of iteration t. 3BASiL's Theorem 1 assumes a nondecreasing positive penalty sequence satisfying $\sum_{t=0}^{\infty} \rho_t^{-1} < \infty$. Our schedule $\rho_t = \rho_0 \eta^{\lfloor t/\kappa \rfloor}$, with $\rho_0 > 0$, $\eta > 1$, and a fixed integer $\kappa \geq 1$, satisfies these conditions, since

$$\sum_{t=1}^{\infty} \frac{1}{\rho_{t-1}} = \frac{\kappa}{\rho_0(1-\eta^{-1})} < \infty.$$

Joint projection. 3BASiL's sparse projection (retaining the k largest nonzero elements) becomes $\Pi_{\mathcal{C}}$, implemented in line 8 of Algorithm 1. For fixed scalar grids, this is exactly QSPROX: nearest-grid rounding minimizes each uncorrected entry's error, and the sparse correction cancels the k largest residuals (Proposition 3.1). Thus, the support-based comparison in 3BASiL's Lemma A.2, Eq. (24), is replaced by

$$\|\mathbf{P}_t - \mathbf{Z}_{t+1}\|_F \leq \|\mathbf{P}_t - \mathbf{Z}_t\|_F, \qquad \mathbf{P}_t = \mathbf{V}_{t+1} + \mathbf{U}_t, \tag{10}$$

since $\mathbf{Z}_t \in \mathcal{C}$ is feasible. This is the only property of the *projection* used in that step.

Low-rank movement. For a fixed finite $C_F \geq 0$, use the gate in line 10 in place of the low-rank movement estimate in 3BASiL's Eq. (18):

$$\|\mathbf{L}_{t+1} - \mathbf{L}_t\|_F \leq C_F \|\mathbf{V}_{t+1} - \mathbf{V}_t\|_F.$$

This movement bound holds for both accepted and rejected candidates by construction. To accommodate for the accept and reject condition, we combine the low-rank norm bound in 3BASiL's Lemma A.1 with the scalar recurrence in Lemma A.3. Write $\mathbf{Y}_t = \rho_t \mathbf{U}_t$, with ρ_t the penalty used at iteration $t+1$, and set

$$a_t = 1 + \|\mathbf{Z}_t\|_F + \frac{\|\mathbf{Y}_t\|_F + \|\mathbf{Y}_{t-1}\|_F}{\rho_{t-1}}, \qquad c = \|\mathbf{H}^{-1/2}\|_2 \|\mathbf{H}^{1/2}\|_2 \max\{1, \|\mathbf{W}\|_F\}.$$

Starting from $E_1 = \max\{a_1, \|\mathbf{L}_1\|_F / c\}$, we prove simultaneously that

$$a_t \leq E_t, \qquad \|\mathbf{L}_t\|_F \leq c E_t, \qquad E_{t+1} = \left(1 + \frac{3C}{\rho_t}\right) E_t,$$

for a sufficiently large fixed constant $C > 0$. Given the bounds at t, the calculation in Lemma A.2, using equation 10, gives $\|\mathbf{Y}_{t+1}\|_F \leq C E_t$ and $\|\mathbf{Z}_{t+1} - \mathbf{Z}_t\|_F \leq 2C E_t / \rho_t$. Hence

$$a_{t+1} \leq a_t + \frac{3C E_t}{\rho_t} \leq E_{t+1}.$$

For an accepted candidate, the weighted-SVD norm bound and $\mathbf{V}_{t+1} = \mathbf{Z}_{t+1} + (\mathbf{Y}_{t+1} - \mathbf{Y}_t)/\rho_t$ give $\|\mathbf{L}_{t+1}\|_F \leq c\, a_{t+1} \leq c E_{t+1}$. For a rejected candidate, $\mathbf{L}_{t+1} = \mathbf{L}_t$, so $\|\mathbf{L}_{t+1}\|_F \leq c E_t \leq c E_{t+1}$. This closes the induction. As in 3BASiL's Lemma A.3, $1 + x \leq e^x$ gives

$$E_t = E_1 \prod_{j=1}^{t-1} \left(1 + \frac{3C}{\rho_j}\right) \leq E_1 \exp\left(3C \sum_{j=1}^{\infty} \rho_j^{-1}\right) < \infty.$$

The established movement bounds therefore give

$$\begin{aligned} \max\left\{\|\mathbf{V}_{t+1} - \mathbf{V}_t\|_F, \|\mathbf{Z}_{t+1} - \mathbf{Z}_t\|_F, \|\mathbf{L}_{t+1} - \mathbf{L}_t\|_F\right\} &\leq K/\rho_{t-1}, \\ \|\mathbf{V}_{t+1} - \mathbf{Z}_{t+1}\|_F \leq K/\rho_t &\leq K/\rho_{t-1}. \end{aligned} \tag{11}$$

The bounds hold for all sufficiently large t.

Thus, the first bound makes $\{\mathbf{V}_t\}$, $\{\mathbf{Z}_t\}$, and $\{\mathbf{L}_t\}$ Cauchy sequences. The second bound makes the consensus residual vanish (consensus is achieved), so $\mathbf{V}_t$ and $\mathbf{Z}_t$ have a common limit $\bar{\mathbf{Z}}$, and $\mathbf{L}_t \to \bar{\mathbf{L}}$.

Feasibility of the limit. For any fixed support T, the matrices that vanish outside T form a closed coordinate subspace. There are finitely many supports of size at most k, so their finite union Σ_k is closed. Additionally, with $\mathbf{Q}_b$ specified with *fixed and finite* grid and candidate matrices having a finite $d_{\text{in}}d_{\text{out}}$ number of entries, the set $\mathcal{Q}_b$ is also finite. Consequently,

$$\mathcal{C} = \mathcal{Q}_b + \Sigma_k = \bigcup_{\mathbf{Q}\in\mathcal{Q}_b} (\mathbf{Q} + \Sigma_k)$$

is a finite union of closed sets. Moreover, the set $\{\mathbf{L} : \operatorname{rank}(\mathbf{L}) \leq r\}$ is also closed. Every iterate satisfies $\mathbf{Z}_t \in \mathcal{C}$ and $\operatorname{rank}(\mathbf{L}_t) \leq r$, so their limits satisfy the same constraints. In particular, there exist $\bar{\mathbf{Q}} \in \mathcal{Q}_b$ and $\bar{\mathbf{S}}$ with $\|\bar{\mathbf{S}}\|_0 \leq k$ such that $\bar{\mathbf{Z}} = \bar{\mathbf{Q}} + \bar{\mathbf{S}}$. Therefore, the reconstructed matrices converge to the feasible decomposition $\bar{\mathbf{Q}} + \bar{\mathbf{S}} + \bar{\mathbf{L}}$.

The conclusion does not assert the optimality of the limit. The guarantee also does not cover the approximate $\mathbf{Q}+\mathbf{S}$ projection used in vector quantization formats like the E8 codebook. That said, the algorithm seems to converge for E8 codebook with experimental results shown in subsection C.4.

C Additional Experiments

C.1 Joint versus sequential fitting with a symmetric scalar quantizer

We extend subsection 4.3 with the full comparison in Table 4. The main comparison fixes the quantization grid, sparse-entry count, and rank; the additional GPTQ (Frantar et al., 2023) and ADMMQ (Lucas et al., 2026) references use $\mathbf{Q}$ alone.

Sequential baseline definitions. `QS_ADMM+LR` fits $\mathbf{Q}+\mathbf{S}$ using rank-zero ADMM, then freezes them and sets $\mathbf{L} = \texttt{wSVD}_r(\mathbf{W}-\mathbf{Q}-\mathbf{S}, \mathbf{H})$. `LR+QS_ADMM` reverses the order: it first sets $\mathbf{L} = \texttt{wSVD}_r(\mathbf{W}, \mathbf{H})$, then freezes $\mathbf{L}$ and fits $\mathbf{Q}+\mathbf{S}$ to $\mathbf{W}-\mathbf{L}$ using rank-zero ADMM. We introduce these two baselines to isolate the gains from the *joint* optimization provided by `QSLR`: they use the same ADMM solver for $\mathbf{Q}+\mathbf{S}$ and the same `wSVD` operation for $\mathbf{L}$, but fit them in separate stages rather than allowing all three components to adapt together.

`Best_seq` evaluates all six orders $\pi \in \mathrm{Perm}(\mathrm{GPTQ}, \mathrm{SparseGPT}, \texttt{wSVD})$. For each weight matrix, Algorithm 2 accumulates the fitted components in $\mathrm{RES} = \mathbf{Q}+\mathbf{S}+\mathbf{L}$, applying each solver to the remaining residual $\mathbf{W}-\mathrm{RES}$. Here `wSVD` is the activation-aware low-rank fit defined in Equation (5); the solvers use the prescribed quantization, sparse-entry, and rank budgets, respectively, with the same damped calibration Hessian $\mathbf{H}$. We fix one of the six permutations, and use the same order of fitting for every layer in a complete model run, and report then the lowest WT2 perplexity across the six runs.

Algorithm 2 `Best_seq`: residual fitting of a layer for one candidate order π.

Input: weight matrix $\mathbf{W}$, damped Hessian $\mathbf{H}$, and fitting order π.
RES $\leftarrow \mathbf{0}$
for compression_func in $\pi(\mathrm{GPTQ}, \mathrm{SparseGPT}, \texttt{wSVD})$ **do**
    RES $\leftarrow$ RES + compression_func($\mathbf{H}, \mathbf{W}-$ RES)
return RES

Table 4: Symmetric scalar quantization: WT2 perplexity (↓). Configurations give $(\mathbf{Q}, \mathrm{sp}, \mathrm{rk})$. `QS` abbreviates `QS_ADMM`. The four correction-based methods use the completed reruns. ADMMQ and GPTQ are quantization-only references, without sparse or low-rank corrections.

Model	Configuration	ADMMQ	GPTQ	`Best_seq`	`QS+LR`	`LR+QS`	`QSLR`
Llama-3.2-1B	(W2, 1%, 16)	251.36	1436.64	41.61	18.98	21.25	**15.41**
	(W2, 1%, 32)	251.36	1436.64	36.27	18.43	19.44	**14.56**
	(W2, 5%, 16)	251.36	1436.64	27.47	13.81	13.83	**12.80**
	(W2, 5%, 32)	251.36	1436.64	26.09	13.77	13.62	**12.50**
	(W4, 1%, 16)	11.21	13.32	10.56	10.10	10.10	**10.05**
	(W4, 1%, 32)	11.21	13.32	10.50	10.09	10.08	**10.02**
	(W4, 5%, 16)	11.21	13.32	10.39	9.94	9.93	**9.92**
	(W4, 5%, 32)	11.21	13.32	10.32	9.94	9.94	**9.91**
Qwen3-4B-Base	(W2, 1%, 16)	25.41	38.14	13.77	10.63	10.93	**9.77**
	(W2, 1%, 32)	25.41	38.14	13.52	10.56	10.83	**9.53**
	(W2, 5%, 16)	25.41	38.14	11.37	9.19	9.24	**8.90**
	(W2, 5%, 32)	25.41	38.14	11.34	9.16	9.17	**8.82**
	(W4, 1%, 16)	8.53	8.57	8.18	8.05	8.06	**8.02**
	(W4, 1%, 32)	8.53	8.57	8.15	8.04	8.04	**8.01**
	(W4, 5%, 16)	8.53	8.57	8.11	7.98	**7.97**	7.98
	(W4, 5%, 32)	8.53	8.57	8.11	7.98	**7.97**	**7.97**

Dense FP16: Llama-3.2-1B WT2 = 9.75; Qwen3-4B-Base WT2 = 7.90.

C.2 JOINT FITTING IN THE SPQR FORMAT

We next replace the separate quantization and sparsity solvers with SpQR, a specialized solver for $\mathbf{Q} + \mathbf{S}$. `SpQR+LR` adds a `wSVD` correction to the frozen SpQR residual; `LR+SpQR` fits `wSVD` first and applies SpQR to the residual. `QSLR` instead fits all three components jointly.

C.2.1 JOINT VERSUS SEQUENTIAL FITTING WITH GROUP16

Full results for Table 2, including Llama-3.2-1B. We use SpQR $(b, 16, 3, 16)$ with τ=1 and ranks $r \in \{0, 8, 16\}$. SpQR selects outliers using the threshold τ rather than a prescribed count. For a fair comparison, `QSLR` retains SpQR's quantization format and measured sparse-entry count in each matrix.

WT2 evaluation uses 2048-token test windows. KL is $\mathrm{KL}(\mathrm{dense} \| \mathrm{compressed}) \times 10^3$, averaged per predicted token at temperature one. Compress (min) reports the compression time measures in minutes. Llama-3.2-1B and Qwen3-4B-Base are reported using a L40S GPU; and Qwen3.8-27B is reported using an H100 GPU.

Table 5: Llama-3.2-1B: SpQR group16 fitting. WT2 perplexity, KL ($\times 10^3$), storage, and compression runtime.

b	r	Method	BPW	WT2 PPL	WT2 KL	Compress (min)
2	0	`SpQR`	2.96	16.89	540.22	1.68
2	0	`QSLR`	2.96	**16.33**	**513.12**	4.89
2	8	`LR+SpQR`	3.05	16.47	521.22	1.83
2	8	`SpQR+LR`	3.05	16.81	540.12	1.83
2	8	`QSLR`	3.05	**15.39**	**450.58**	6.26
2	16	`LR+SpQR`	3.14	16.01	493.74	1.87
2	16	`SpQR+LR`	3.15	16.74	531.58	1.86
2	16	`QSLR`	3.15	**14.88**	**421.37**	6.44
3	0	`SpQR`	3.64	10.78	100.35	1.56
3	0	`QSLR`	3.64	**10.73**	**95.03**	4.65
3	8	`LR+SpQR`	3.73	10.75	96.25	1.90
3	8	`SpQR+LR`	3.73	10.75	98.14	1.94
3	8	`QSLR`	3.73	**10.63**	**86.06**	6.28
3	16	`LR+SpQR`	3.82	10.72	95.26	1.92
3	16	`SpQR+LR`	3.82	10.78	98.35	1.89
3	16	`QSLR`	3.82	**10.59**	**82.34**	6.40
—	—	Dense	—	9.75	0.00	—

Table 6: Qwen3-4B-Base: SpQR group16 fitting. WT2 perplexity, KL ($\times 10^3$), storage, and compression runtime.

b	r	Method	BPW	WT2 PPL	WT2 KL	Compress (min)
2	0	`SpQR`	2.98	10.14	292.33	5.04
2	0	`QSLR`	2.98	**9.81**	**259.20**	18.49
2	8	`LR+SpQR`	3.05	10.23	298.50	6.12
2	8	`SpQR+LR`	3.05	10.18	294.65	6.35
2	8	`QSLR`	3.05	**9.72**	**247.20**	25.63
2	16	`LR+SpQR`	3.12	10.18	293.09	5.58
2	16	`SpQR+LR`	3.12	10.12	289.39	5.60
2	16	`QSLR`	3.12	**9.55**	**230.04**	24.37
3	0	`SpQR`	3.64	8.35	69.35	4.82
3	0	`QSLR`	3.64	**8.33**	**65.77**	18.47
3	8	`LR+SpQR`	3.71	8.29	62.35	5.87
3	8	`SpQR+LR`	3.71	8.31	65.92	5.91

b	r	Method	BPW	WT2 PPL	WT2 KL	Compress (min)
3	8	QSLR	3.71	**8.22**	**56.98**	24.83
3	16	LR+SpQR	3.78	8.30	62.09	5.67
3	16	SpQR+LR	3.79	8.30	64.46	5.65
3	16	QSLR	3.79	**8.23**	**54.97**	25.08
—	—	Dense	—	7.90	0.00	—

Table 7: Qwen3.8-27B: SpQR group16 fitting. WT2 perplexity, KL ($\times 10^3$), storage, and compression runtime.

b	r	Method	BPW	WT2 PPL	WT2 KL	Compress (min)
2	0	SpQR	2.95	**7.91**	162.25	20.16
2	0	QSLR	2.95	7.97	**157.30**	115.40
2	8	LR+SpQR	2.98	7.81	159.93	25.91
2	8	SpQR+LR	2.98	7.85	160.43	26.01
2	8	QSLR	2.98	**7.77**	**149.36**	147.72
2	16	LR+SpQR	3.02	**7.85**	158.54	26.61
2	16	SpQR+LR	3.02	7.89	160.69	26.71
2	16	QSLR	3.02	7.97	**152.86**	149.53
3	0	SpQR	3.63	7.28	46.09	21.94
3	0	QSLR	3.63	**7.21**	**43.14**	116.24
3	8	LR+SpQR	3.67	7.25	44.84	24.78
3	8	SpQR+LR	3.67	7.26	44.37	24.48
3	8	QSLR	3.67	**7.21**	**41.99**	145.62
3	16	LR+SpQR	3.71	7.23	43.79	24.52
3	16	SpQR+LR	3.71	7.26	44.51	24.23
3	16	QSLR	3.71	**7.23**	**41.94**	146.82
—	—	Dense	—	7.05	0.00	—

Table 8 compares allocations in the SpQR $(2, 64, 3, 16)$ format on Qwen3-4B-Base. The QSLR rows share a 2.81-BPW budget.

Table 8: Qwen3-4B-Base, SpQR(2,64,3,16) format. WT2 perplexity (↓)

Method	BPW	WT2
SpQR ($\mathbf{Q} + \mathbf{S}$)	2.81	12.65
QSLR $\mathbf{Q} + \mathbf{S}$	2.81	11.43
QSLR $\mathbf{Q} + \mathbf{S} + \mathbf{L}$, $r = 16$	2.81	10.92
QSLR $\mathbf{Q} + \mathbf{S} + \mathbf{L}$, $r = 32$	2.81	10.66
QSLR $\mathbf{Q} + \mathbf{S} + \mathbf{L}$, $r = 56$	2.81	**10.47**
QSLR $\mathbf{Q} + \mathbf{L}$	2.81	10.88
SpQR+LR, $r = 16$	2.95	12.52
SpQR+LR, $r = 32$	3.10	12.42
SpQR+LR, $r = 56$	3.31	12.25
Qwen3-4B-Base dense FP16: WT2 = 7.90.		

C.3 SPARSITY VS. LOW-RANK TRADEOFFS

These additional experiments complement subsection 4.5, varying the allocation of a fixed correction budget between sparse and low-rank components.

C.3.1 SYMMETRIC SCALAR QUANTIZATION TRADEOFFS

The W2/W3 experiments use symmetric per-channel grids. Correction storage matches rank 16 plus 1% sparsity within each model. Rank zero is $\mathbf{Q} + \mathbf{S}$; the no-sparsity endpoint is $\mathbf{Q} + \mathbf{L}$.

Table 9: Symmetric scalar W2/W3 fixed-correction-budget comparisons. WT2 perplexity (↓).

Llama-3.2-1B

Format	Method	BPW	WT2
W2	QSLR r0 ($\mathbf{Q} + \mathbf{S}$)	2.51	20.01
W2	QSLR r8	2.51	15.45
W2	QSLR r16	2.51	**15.20**
W2	QSLR r24	2.51	15.29
W2	QSLR r32	2.51	15.52
W2	QSLR $\mathbf{Q} + \mathbf{L}$	2.51	17.92
W3	QSLR r0 ($\mathbf{Q} + \mathbf{S}$)	3.51	11.04
W3	QSLR r8	3.51	10.95
W3	QSLR r16	3.51	10.92
W3	QSLR r24	3.51	10.92
W3	QSLR r32	3.51	**10.89**
W3	QSLR $\mathbf{Q} + \mathbf{L}$	3.51	11.13
FP16	Dense model	16	9.75

Qwen3-4B-Base

Format	Method	BPW	WT2
W2	QSLR r0 ($\mathbf{Q} + \mathbf{S}$)	2.47	10.59
W2	QSLR r8	2.47	**9.76**
W2	QSLR r16	2.47	**9.76**
W2	QSLR r24	2.47	9.81
W2	QSLR r32	2.47	9.81
W2	QSLR r48	2.47	10.02
W2	QSLR $\mathbf{Q} + \mathbf{L}$	2.47	10.77
W3	QSLR r0 ($\mathbf{Q} + \mathbf{S}$)	3.47	8.35
W3	QSLR r8	3.47	**8.34**
W3	QSLR r16	3.47	**8.34**
W3	QSLR r24	3.47	8.35
W3	QSLR r32	3.47	8.38
W3	QSLR r48	3.47	8.35
W3	QSLR $\mathbf{Q} + \mathbf{L}$	3.47	8.48
FP16	Dense model	16	7.90

Table 10: Llama-3-8B: symmetric allocation at fixed BPW, using `QSLR`. WT2 perplexity (↓).

Symmetric W2, 2.42 BPW

Allocation	Rank	Sp. (%)	WT2	Compress (min)
$\mathbf{Q} + \mathbf{S}$	0	1.30	18.364	22.96
$\mathbf{Q} + \mathbf{S} + \mathbf{L}$	16	1.00	**8.590**	38.59
$\mathbf{Q} + \mathbf{S} + \mathbf{L}$	32	0.70	8.608	40.07
$\mathbf{Q} + \mathbf{S} + \mathbf{L}$	54	0.29	8.751	42.25
$\mathbf{Q} + \mathbf{L}$	69	0.00	9.409	42.38
Dense FP16	—	—	6.136	—

Symmetric W3, 3.42 BPW

Allocation	Rank	Sp. (%)	WT2	Compress (min)
$\mathbf{Q} + \mathbf{S}$	0	1.30	6.808	23.03
$\mathbf{Q} + \mathbf{S} + \mathbf{L}$	16	1.00	6.788	37.84
$\mathbf{Q} + \mathbf{S} + \mathbf{L}$	32	0.70	**6.759**	40.68
$\mathbf{Q} + \mathbf{S} + \mathbf{L}$	54	0.29	6.767	42.87
$\mathbf{Q} + \mathbf{L}$	69	0.00	6.861	43.04
Dense FP16	—	—	6.136	—

C.3.2 SPQR-FORMAT ALLOCATION TRADEOFFS

Tables 11 and 12 group the allocation experiments by SpQR format. Each model keeps BPW fixed as storage shifts from $\mathbf{S}$ to $\mathbf{L}$.

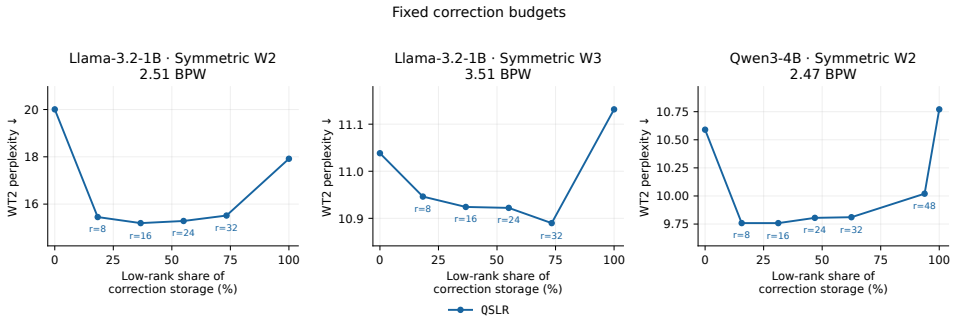

Figure 4: Symmetric W2/W3 rank comparisons on Llama-1B and W2 on Qwen3-4B under fixed BPW.

Table 11: SpQR $(2, 16, 3, 16)$ allocation comparisons at fixed BPW, using `QSLR` throughout. Bold marks the lowest perplexity within each model and format.

SpQR $(2, 16, 3, 16)$

Llama-3.2-1B 2.96 BPW		**Qwen3-4B-Base** 2.98 BPW		**Qwen3.8-27B** 2.95 BPW	
Allocation	WT2 ↓	Allocation	WT2 ↓	Allocation	WT2 ↓
$\mathbf{Q}+\mathbf{S}$	16.23	$\mathbf{Q}+\mathbf{S}$	9.82	$\mathbf{Q}+\mathbf{S}$	7.87
$\mathbf{Q}+\mathbf{S}+\mathbf{L}$, $r=8$	15.69	$\mathbf{Q}+\mathbf{S}+\mathbf{L}$, $r=8$	**9.62**	$\mathbf{Q}+\mathbf{S}+\mathbf{L}$, $r=8$	7.98
$\mathbf{Q}+\mathbf{S}+\mathbf{L}$, $r=16$	**15.36**	$\mathbf{Q}+\mathbf{S}+\mathbf{L}$, $r=16$	9.66	$\mathbf{Q}+\mathbf{S}+\mathbf{L}$, $r=16$	**7.82**
$\mathbf{Q}+\mathbf{S}+\mathbf{L}$, $r=24$	15.58	$\mathbf{Q}+\mathbf{S}+\mathbf{L}$, $r=24$	9.65	$\mathbf{Q}+\mathbf{S}+\mathbf{L}$, $r=24$	7.93
$\mathbf{Q}+\mathbf{L}$	19.72	$\mathbf{Q}+\mathbf{L}$	12.13	$\mathbf{Q}+\mathbf{L}$	8.07
Dense FP16	9.75	Dense FP16	7.90	Dense FP16	7.05

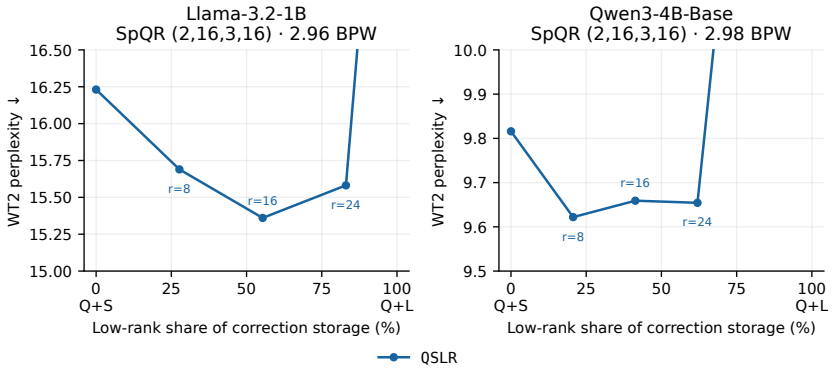

Figure 5: SpQR $(2, 16, 3, 16)$ allocation curves for Llama-3.2-1B and Qwen3-4B-Base in Table 11. Lines connect measured allocations at fixed BPW.

Table 12: Qwen3-4B-Base: SpQR group64 and group128 allocations at fixed BPW.

SpQR $(2, 64, 3, 16)$
Qwen3-4B-Base
2.81 BPW

Allocation	WT2 ↓
$\mathbf{Q} + \mathbf{S}$	11.43
$\mathbf{Q} + \mathbf{S} + \mathbf{L}$, $r = 16$	10.92
$\mathbf{Q} + \mathbf{S} + \mathbf{L}$, $r = 32$	10.66
$\mathbf{Q} + \mathbf{S} + \mathbf{L}$, $r = 56$	**10.47**
$\mathbf{Q} + \mathbf{L}$	10.88
Dense FP16	7.90

SpQR $(2, 128, 3, 16)$
Qwen3-4B-Base
2.74 BPW

Allocation	WT2 ↓
$\mathbf{Q} + \mathbf{S}$	13.89
$\mathbf{Q} + \mathbf{S} + \mathbf{L}$, $r = 40$	**11.96**
$\mathbf{Q} + \mathbf{L}$	12.19
Dense FP16	7.90

C.4 E8 COMPRESSION AND ALGORITHM ABLATIONS

We show the procedure is quantizer-agnostic by integrating the E8 lattice codebook introduced in QuIP# (Tseng et al., 2024). Rather than rounding each weight independently, this vector quantizer encodes a group of eight weights by an index into a finite codebook derived from the eight-dimensional E_8 lattice. The W3 format uses residual vector quantization to encode a correction to the first codeword.

Randomized Hadamard rotations are enabled by default in all E8 experiments. Here a rotation means an orthogonal change of coordinates applied before compression. The structured transform can be applied efficiently without storing a dense rotation matrix. Its purpose is to spread large entries across coordinates, making the transformed weights better suited to the E8 codebook (Tseng et al., 2024). For the convention $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, let $\mathbf{R}_{\text{in}}$ and $\mathbf{R}_{\text{out}}$ denote the fixed orthogonal transforms and let $s > 0$ be the weight normalization scale. We fit the decomposition in the coordinates

$$\widetilde{\mathbf{W}} = \mathbf{R}_{\text{in}}^{\top}(\mathbf{W}/s)\mathbf{R}_{\text{out}}, \qquad \widetilde{\mathbf{H}} = \mathbf{R}_{\text{in}}^{\top}\mathbf{H}\mathbf{R}_{\text{in}}. \tag{12}$$

We store the E8 code indices, sparse entries, and low-rank factors fitted in these coordinates, together with the random signs and scales of the Hadamard rotation. We use the same preprocessing for the matched CALDERA comparison (Saha et al., 2024).

Allocation between sparsity and low rank. There is no consistent preference for sparse, low-rank, or mixed corrections across the studied E8 settings. For example, Llama-3.2-1B E8 W2 comparison favors $\mathbf{Q} + \mathbf{L}$, while Qwen3-4B E8 W2 favors $\mathbf{Q} + \mathbf{S}$; with the mixed representtion remaining competitive. We hypothesize that the Hadamard rotations contribute to this behavior as spreading large entries across coordinates changes the residual structure seen by the quantizer and can reduce the benefit of the correction components.

Comparison with CALDERA. At matched correction storage, our $\mathbf{Q} + \mathbf{L}$ solver achieves lower WT2 perplexity than CALDERA in all four Llama-3.2-1B and Qwen3-4B settings in Table 13, while being faster to optimize the allocation. In our experiments, CALDERA performs 20 alternating-minimization iterations per weight matrix, each updating the low-rank factors followed by the quantized component. Each quantization update uses Block LDLQ to quantize the residual block by block with Hessian-based error compensation, followed by ten refinement passes that revisit all blocks using the current reconstruction error. In contrast, `QSLR` separates the Hessian-aware consensus solve from a batched fixed-codebook projection and uses up to 300 ADMM iterations with early stopping. The number of iterations of our algorithm might be higher, but they are much more efficient subroutines.

Table 13: E8 compression with fixed correction storage and Hadamard rotations. Compress (min) is measured on L40S for Llama-3.2-1B and H100 for Qwen3-4B-Base.

Llama-3.2-1B

Format	Method	BPW	WT2	Compress (min)
W2	QuIP# (Q only)	2.01	21.34	13.2
W2	QSLR r0 ($\mathbf{Q} + \mathbf{S}$)	2.53	15.38	48.1
W2	QSLR r8	2.53	15.21	50.4
W2	QSLR r16	2.53	14.74	52.0
W2	QSLR r24	2.53	14.65	47.6
W2	QSLR r32	2.53	14.46	46.4
W2	QSLR $\mathbf{Q} + \mathbf{L}$	2.53	**14.35**	36.3
W2	CALDERA $\mathbf{Q} + \mathbf{L}$	2.53	16.34	194.5
W3	QuIP# (Q only)	3.01	11.50	12.1
W3	QSLR r0 ($\mathbf{Q} + \mathbf{S}$)	3.53	11.03	56.2
W3	QSLR r8	3.53	10.96	57.2
W3	QSLR r16	3.53	10.92	55.4
W3	QSLR r24	3.53	10.91	56.4
W3	QSLR r32	3.53	10.86	53.9
W3	QSLR $\mathbf{Q} + \mathbf{L}$	3.53	**10.77**	40.5
W3	CALDERA $\mathbf{Q} + \mathbf{L}$	3.53	11.03	205.6
FP16	Dense model	16	9.75	—

Qwen3-4B-Base

Format	Method	BPW	WT2	Compress (min)
W2	QuIP# (Q only)	2.01	10.55	34.8
W2	QSLR r0 ($\mathbf{Q} + \mathbf{S}$)	2.48	**9.43**	182.0
W2	QSLR r8	2.48	9.70	187.4
W2	QSLR r16	2.48	9.67	179.5
W2	QSLR r24	2.48	9.64	178.0
W2	QSLR r32	2.48	9.60	179.7
W2	QSLR r48	2.48	9.62	152.9
W2	QSLR $\mathbf{Q} + \mathbf{L}$	2.48	9.55	131.7
W2	CALDERA $\mathbf{Q} + \mathbf{L}$	2.48	9.89	578.8
W3	QuIP# (Q only)	3.01	8.45	36.4
W3	QSLR r0 ($\mathbf{Q} + \mathbf{S}$)	3.48	8.28	203.8
W3	QSLR r8	3.48	8.28	210.3
W3	QSLR r16	3.48	8.25	202.4
W3	QSLR r24	3.48	8.25	198.2
W3	QSLR r32	3.48	8.25	196.1
W3	QSLR r48	3.48	**8.24**	176.6
W3	QSLR $\mathbf{Q} + \mathbf{L}$	3.48	8.25	145.6
W3	CALDERA $\mathbf{Q} + \mathbf{L}$	3.48	8.33	624.3
FP16	Dense model	16	7.90	—

C.5 LoftQ's Data-free NF2 decomposition ($\mathbf{Q} + \mathbf{L}$) and LoRA

LoftQ is a data-free (we can think of data-free methods as setting $\mathbf{X} = \mathbf{I}$ in Equation (1)) specialized solver for $\mathbf{Q} + \mathbf{L}$ compression. We compare LoftQ and `QSLR` (with sparsity set to 0 or and at matched BPW and under a data-free setting; hence we do not use any calibration data points for our method in these experiments) on Meta-Llama-3-8B using NF2/block64; here NF2 is the NormalFloat quantization format introduced by Dettmers et al. (2024). LoftQ uses five alternating updates with scale refitting; whereas our solver `QSLR` fixes scales after initialization as described in Section A. For both methods, the low-rank factors initialize the LoRA components and are the only trainable parameters during the LoRA finetuning stage. All experiments use 100 AdamW steps on 204,800 WikiText-2 training tokens, learning rate 5×10^{-5}.

Tables 14 and 15 report post-LoRA perplexity and zero-shot accuracy, respectively. The (+1%) rows add additional sparse correction budget. We evaluate these decompositions as initializations for LoRA.

Table 14: Meta-Llama-3-8B perplexity after 100-step LoRA. Params.(M) counts trainable LoRA parameters in the millions. Compress (min) is the compression runtime on L40S GPU in minutes.

Method	r	BPW	Params.(M)	Step	WT2	C4	PTB	Compress (min)
LoftQ NF2	16	2.60	41.94	100	20.79	54.86	67.20	32.46
QSLR NF2 $\mathbf{Q} + \mathbf{L}$	16	2.60	41.94	100	14.70	34.45	43.33	10.51
QSLR NF2 $\mathbf{Q} + \mathbf{S} + \mathbf{L}$ (matched)	8	2.60	20.97	100	14.73	34.60	40.31	11.53
QSLR NF2 $\mathbf{Q} + \mathbf{S} + \mathbf{L}$ (+1%)	16	3.08	41.94	100	**12.10**	**26.70**	**32.42**	11.80
LoftQ NF2	64	2.88	167.77	100	17.35	44.30	56.06	32.54
QSLR NF2 $\mathbf{Q} + \mathbf{L}$	64	2.88	167.77	100	12.71	29.94	32.77	13.07
QSLR NF2 $\mathbf{Q} + \mathbf{S} + \mathbf{L}$ (matched)	32	2.88	83.89	100	12.20	27.14	33.28	12.74
QSLR NF2 $\mathbf{Q} + \mathbf{S} + \mathbf{L}$ (+1%)	64	3.36	167.77	100	**10.80**	**22.74**	**28.33**	14.50
Llama-3-8B dense FP16 (16 BPW): WT2/C4/PTB $= 6.14/9.45/11.18$.								

Table 15: Meta-Llama-3-8B post-LoRA zero-shot accuracy (%). Avg. is the unweighted mean over all eight tasks.

Method	r	BPW	PIQA	ARC-E	ARC-C	Hella.	Wino.	OBQA	BoolQ	RTE	Avg.
LoftQ NF2	16	2.60	63.71	40.51	27.22	41.41	54.66	30.10	49.65	52.17	44.93
QSLR NF2 $\mathbf{Q} + \mathbf{L}$	16	2.60	66.65	46.70	29.65	50.40	57.26	31.80	62.51	50.72	49.46
QSLR NF2 $\mathbf{Q} + \mathbf{S} + \mathbf{L}$ (matched)	8	2.60	66.73	46.49	29.18	50.81	58.13	33.90	64.01	54.87	50.51
QSLR NF2 $\mathbf{Q} + \mathbf{S} + \mathbf{L}$ (+1%)	16	3.08	**70.32**	**54.27**	**33.96**	**58.46**	**59.23**	**35.40**	**66.35**	**56.14**	**54.27**
LoftQ NF2	64	2.88	65.13	41.90	26.15	45.25	54.78	30.50	62.31	51.44	47.18
QSLR NF2 $\mathbf{Q} + \mathbf{L}$	64	2.88	68.55	52.67	31.95	55.14	60.73	33.00	59.07	53.43	51.82
QSLR NF2 $\mathbf{Q} + \mathbf{S} + \mathbf{L}$ (matched)	32	2.88	70.27	55.66	34.13	58.03	59.87	35.10	**64.97**	52.71	53.84
QSLR NF2 $\mathbf{Q} + \mathbf{S} + \mathbf{L}$ (+1%)	64	3.36	**71.49**	**59.32**	**36.43**	**62.21**	**62.27**	**36.20**	64.59	**57.22**	**56.22**

Effect of sparsity. Adding 1% sparsity to our $\mathbf{Q} + \mathbf{L}$ initialization increases mean post-LoRA eight-task accuracy from 49.46% to 54.27% at rank 16 and from 51.82% to 56.22% at rank 64, gains of 4.81 and 4.40 percentage points. The sparse component adds 0.48 BPW, while the number of trainable factors and the 100-step training budget remain unchanged. At matched BPW, $\mathbf{Q}+\mathbf{S}+\mathbf{L}$ reaches 50.51% versus 49.46% and 53.84% versus 51.82%, respectively, while halving the number of trainable parameters.

D HYPERPARAMETER ABLATIONS

We vary the number of calibration data points, the low-rank acceptance threshold C_F, and Hessian damping ρ one at a time on Llama-3.2-1B using symmetric per-channel 2-bit quantization with 1% sparsity and rank 16.

The results seem to be robust by changing these hyperparameters. In our experiments, these values by default are: 128 calibration data points of 2048 tokens, $C_F = 0.3$, $\rho = 0.1$.

Table 16: Ablation study of ADMM parameters on Llama-3.2-1B perplexity (↓).

Hyperparameter	Value	WT2	C4	PTB
Calibration points	32	15.88	30.61	32.91
	64	15.50	28.96	32.24
	128	15.40	29.67	31.42
	256	15.12	28.36	31.07
	512	15.19	28.48	31.88
C_F	0.1	17.13	34.44	35.34
	0.3	15.27	28.61	31.33
	1.0	15.28	29.67	30.90
	10.0	15.26	28.47	30.95
	∞ (always accept)	15.26	28.47	30.95
Damping	0.01	15.46	31.11	33.32
	0.1	15.27	28.61	31.33
	1.0	16.34	29.89	33.37

We repeat the same ablations on a much larger model Qwen3.8-27B with the same settings; results are reported in Table 17.

Table 17: Ablation study of ADMM parameters on Qwen3.8-27B perplexity (↓).

Hyperparameter	Value	WT2	C4	PTB
Calibration points	32	8.01	13.32	12.45
	64	7.96	13.21	12.27
	128	7.94	13.12	12.23
	256	7.83	13.07	12.06
	512	7.82	13.04	11.99
C_F	0.1	8.11	13.46	12.58
	0.3	7.94	13.12	12.23
	1.0	7.92	13.11	12.17
	10.0	7.92	13.11	12.17
	∞ (always accept)	7.97	13.12	12.17
Damping	0.01	7.98	13.26	12.24
	0.1	7.94	13.12	12.23
	1.0	8.02	13.04	12.24

E GPU KERNELS ADDITIONAL RESULTS

These are additional GPU acceleration results of our mixed-representation compression that complement our results in Table 3 under the same setting. Only the sparsity pattern is changed to include 0.1% and 0.5%.

Table 18: Kernel matrix mulitplication speedups over dense FP16 at two sparse-correction densities.

	Sparsity 0.1%		Sparsity 0.5%	
Dimensions ($m \times n$)	CUDA cores	Tensor cores	CUDA cores	Tensor cores
4096×4096	$0.734\times$	$0.659\times$	$0.721\times$	$0.589\times$
4096×11008	$2.325\times$	$2.572\times$	$2.227\times$	$2.217\times$
11008×4096	$2.108\times$	$2.462\times$	$2.023\times$	$2.198\times$
16384×16384	$2.512\times$	$2.892\times$	$2.403\times$	$2.726\times$