REALIGNQ: POST-COMPRESSION REFINEMENT FOR JOINT SPARSIFICATION AND QUANTIZATION OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantization and sparsification compress large language models by restricting numerical precision and nonzero support, respectively. Support selected in continuous weight space may be poorly matched to the values that its retained coordinates can realize on the quantization grid. Moreover, criteria applied to individual coordinates overlook interactions among simultaneous changes in support membership and quantized values. We therefore propose ReAlignQ, a refinement framework that jointly selects support exchanges and legal discrete values for incoming weights under fixed quantization scales and a prescribed sparsity budget. For each linear weight matrix, ReAlignQ models interactions among screened candidates using an empirical gradient Gram surrogate built from gradients of individual calibration samples. It retains the resulting gradient signatures only for screened candidates, which reduces the storage needed to model their interactions. We further implement sparse and quantized operators with exact bitmap masks. These operators execute refined nonuniform masks without changing them. We also provide theoretical results that characterize support selection over legal codes, bound the empirical gradient Gram interactions omitted by diagonal screening, and establish surrogate descent while preserving sparsity for accepted exchanges. Across Llama and Qwen models and multiple sparsity levels, ReAlignQ consistently improves language-modeling quality without retraining, changing parent scales, or relaxing sparsity budgets. On Llama 3 8B with an EIWR W8A16 parent at 70% sparsity, it reduces WikiText-2 perplexity from 35.48 to 25.57 and improves mean zero-shot accuracy by 2.6 percentage points.

1 INTRODUCTION

Large language models deliver strong performance but incur substantial storage, memory traffic, and computation costs during inference. Quantization and sparsification are therefore widely used to reduce their deployment cost (Frantar et al., 2023; Frantar & Alistarh, 2023; Ashkboos et al., 2024b;a; Lin et al., 2025).

Quantization reduces numerical precision by mapping continuous parameters to low-bit representations (Xiao et al., 2023; Shao et al., 2024; Egiazarian et al., 2024; Sun et al., 2025), whereas sparsification limits the nonzero support to reduce storage and computation (Frantar & Alistarh, 2023; Sun et al., 2024; Zhang et al., 2024a;b). We use *code assignment* to refer to the selection of a discrete quantized value for a particular weight. Aggressively reducing the bit width or increasing sparsity, however, often causes a sharp degradation in model quality (Harma et al., 2025; Guo et al., 2024; Mozaffari et al., 2025; Guo et al., 2026; Kim et al., 2024). Combining quantization and sparsification enables more aggressive compression, but also couples the choice of nonzero locations with the discrete values assigned to them (Frantar & Alistarh, 2022; Guo et al., 2024). We focus on a complementary setting that refines an existing model after sparsification and quantization while keeping its quantization scales and prescribed support sizes fixed.

When sparsification and quantization are optimized under separate objectives, their independently obtained local decisions need not jointly constitute the best state under the final compression constraints (Harma et al., 2025). Sparsification typically selects the nonzero support according to parameter importance or reconstruction error in the continuous weight space (Frantar & Alistarh, 2023; Sun et al., 2024; Su et al., 2025), whereas quantization evaluates representation quality on a finite

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.

grid (Frantar et al., 2023; Lin et al., 2024; Shao et al., 2024). This creates two challenges. First, the benefit of allocating one of the limited nonzero slots to a coordinate depends on the legal quantized values that the coordinate can realize (Harma et al., 2025; Guo et al., 2024). A coordinate favored in continuous weight space may therefore be poorly matched to the quantization grid (Dettmers et al., 2024). Second, when multiple support and code changes are applied together, their effects on the calibration loss may reinforce or offset one another and cannot be fully characterized by independent per-coordinate scores (Mozaffari et al., 2025; Guo et al., 2026; Kim et al., 2025; Sieberling et al., 2025). Although recent methods have begun to coordinate these decisions from different perspectives (Guo et al., 2024; Mozaffari et al., 2025; Guo et al., 2026; Zeng et al., 2025), many compression pipelines still rely on separately constructed support-selection criteria and quantization objectives. More generally, jointly evaluating support selection, code assignment, and their interactions at the scale of LLMs remains challenging.

To address this problem, we propose *ReAlignQ*, a refinement framework for models that are both sparse and quantized. Its core idea is to evaluate support selection and code assignment under the same local objective. Rather than making these decisions independently, ReAlignQ evaluates the physical weight changes that they induce on the quantization grid. In each linear weight matrix, the objective accounts for the selected support transitions, their discrete values, and the interactions among them. ReAlignQ estimates these interactions from inner products between gradient signatures obtained from individual calibration samples. Each exchange removes one existing nonzero, activates one previously zero coordinate, and selects a legal quantized code for the incoming coordinate. This preserves the target support size. The incoming code is evaluated together with the outgoing coordinate rather than selected after fixing the support. ReAlignQ restricts this joint evaluation to screened candidates, which makes the refinement practical for LLMs while preserving the target sparsity and quantization constraints.

Beyond the refinement framework itself, we complement ReAlignQ with inference execution support and theoretical analysis. We implement sparse and quantized operators with exact bitmap masks to verify that refined nonuniform support patterns can be executed without altering their supplied masks. We also provide theoretical analysis of its key design choices. Experiments across different model families and sparsity levels show that ReAlignQ effectively mitigates the quality degradation caused by joint sparsification and quantization, without retraining the model, altering the parent quantization scales, or relaxing the prescribed compression budget. The gains are particularly pronounced under aggressive compression, where ReAlignQ improves both language-modeling quality and zero-shot reasoning performance. For example, on Llama3-8B with an EIWR W8A16 parent at 70% sparsity, ReAlignQ reduces WikiText-2 perplexity from 35.48 to 25.57 and improves the mean zero-shot accuracy by 2.6 percentage points.

Our main contributions are summarized as follows:

- We identify a mismatch between support selection and the discrete quantized values that retained weights can realize. Optimizing sparsification and quantization separately need not yield the best state under their joint constraints. We formulate refinement as the coupled selection of support exchanges and legal codes for incoming weights under fixed quantization scales and a prescribed sparsity budget.
- We propose ReAlignQ, which screens candidates while accounting for the quantization grid and selects the incoming code for each candidate exchange.
- We implement sparse and quantized inference operators with exact bitmap masks. We also analyze support selection over legal codes, interaction modeling, and constrained exchange.
- We validate ReAlignQ across multiple model families and sparsity levels. It consistently improves language-modeling quality and recovers zero-shot reasoning performance under aggressive compression.

2 Related Work

Post-Training Quantization. Post-training quantization (PTQ) uses calibration data to reduce the error of low-bit representations without full retraining. Representative approaches use approximate second-order reconstruction (Frantar et al., 2023), activation-aware scaling (Lin et al., 2024), or learned rotations (Liu et al., 2025). They focus on numerical representation for dense or

fixed-support models; ReAlignQ additionally re-evaluates whether a coordinate should occupy the nonzero budget.

One-Shot Pruning. One-shot pruning constructs sparse LLMs without retraining, using approximate reconstruction (Frantar & Alistarh, 2023), activation-aware saliency (Sun et al., 2024), or mask rebuilding with block-aware optimization (Su et al., 2025). Within this setting, EIWR refines pruned weights through training-free, layer-wise continuous optimization (Xie et al., 2026). ReAlignQ instead refines a discrete sparse-quantized state by jointly selecting releases, activations, and their incoming quantized values.

Joint Quantization and Sparsification. Sparsity and quantization are not generally independent operations, and their composition and ordering have been analyzed theoretically and empirically (Harma et al., 2025). Existing LLM frameworks coordinate the two through quantization-friendly sparsity and activation editing (Guo et al., 2024), one-shot compression with low-rank compensation (Mozaffari et al., 2025), second-order group error compensation (Guo et al., 2026), or system–algorithm co-design for group sparsity and low-bit decoding (Zeng et al., 2025). Rather than constructing a new compression pipeline, ReAlignQ starts from an initial sparse-quantized state and searches admissible support-code transitions under fixed quantization scales and target support sizes.

3 Theoretical Analysis

We state three results for the restricted candidate space used by ReAlignQ. All statements concern one linear weight matrix and its screened candidate set $\mathcal{C}$, with coordinates outside $\mathcal{C}$ and quantization scales fixed. Detailed assumptions, proofs, and extensions are deferred to the supplementary material. In this section, z_i^0 denotes the reference code, s_i denotes its fixed scale, and $\mathcal{Q}_i^{\mathrm{adm}}$ denotes the admissible final codes. A final code q induces the physical change $\Delta_i(q) = s_i(q - z_i^0)$. For a matrix with prescribed support size K, let $K_{\mathcal{C}} = K - \|z^0_{\mathcal{C}^c}\|_0$ denote the number of nonzero coordinates required within $\mathcal{C}$.

Proposition 1 (Support Selection over Quantized Codes)**.** *Given gradient g_i, diagonal statistic $F_i \geq 0$, and damping $\lambda \geq 0$, define*

$$\phi_i(q) = g_i\Delta_i(q) + \frac{1}{2}(F_i + 2\lambda)\Delta_i(q)^2, \qquad \delta_i = \min_{q \in \mathcal{Q}_i^{\mathrm{adm}} \setminus \{0\}} \phi_i(q) - \phi_i(0). \tag{1}$$

Under a separable diagonal surrogate, assume that exactly $K_{\mathcal{C}}$ nonzero coordinates are required. An optimum selects the $K_{\mathcal{C}}$ smallest δ_i and assigns their minimizing admissible nonzero codes.

Proposition 1 identifies the correct code-aware support score: it motivates evaluating activation candidates over their realizable quantized codes rather than using a code-oblivious continuous importance.

Theorem 1 (Interaction Bound for Gradient Signatures)**.** *Let $U_i \in \mathbb{R}^N$ be the per-sample gradient signature of candidate i, assume $F_i = \|U_i\|_2^2/N$, and let $S = \mathrm{supp}(\Delta)$ with $|S| = k$. Define*

$$D(\Delta) = \frac{1}{2}\sum_{i \in S} F_i\Delta_i^2, \qquad I(\Delta) = \frac{1}{N}\sum_{\substack{i<j \\ i,j \in S}} U_i^\top U_j\, \Delta_i\Delta_j, \qquad \mu = \max_{\substack{i \neq j \\ \|U_i\|_2\|U_j\|_2 > 0}} \frac{|U_i^\top U_j|}{\|U_i\|_2\|U_j\|_2}, \tag{2}$$

with $\mu = 0$ when fewer than two signatures are nonzero. Then

$$|I(\Delta)| \leq \mu(k-1)D(\Delta). \tag{3}$$

Theorem 1 bounds the empirical gradient-Gram interaction discarded by diagonal evaluation, thereby motivating retention of sample-wise signatures in the screened candidate space. It is not a true-Hessian or true-loss guarantee.

Proposition 2 (Guarantee for Accepted Exchanges)**.** *Fix $\tau \geq 0$. From a feasible state $\|q^{(t)}\|_0 = K$, consider a pair that activates one coordinate that is currently zero, releases one coordinate that is currently nonzero, and whose exact change in the surrogate is below $-\tau$. Then*

$$\|q^{(t+1)}\|_0 = K, \qquad J\big(\Delta^{(t+1)}\big) < J\big(\Delta^{(t)}\big) - \tau. \tag{4}$$

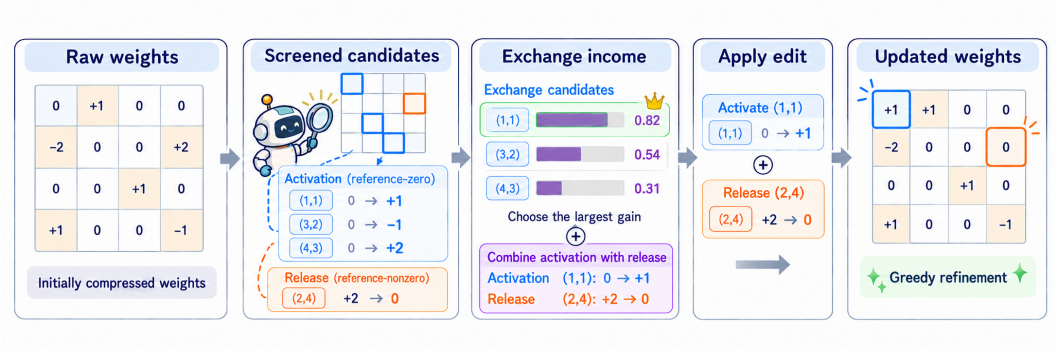

Figure 1: **Overview of ReAlignQ.** Starting from initially compressed weights, ReAlignQ screens zero entries for activation and nonzero entries for release. It ranks joint candidate exchanges by exchange income and selects an activation coordinate, a release coordinate, and the incoming quantized value together. The selected edit activates one zero entry and releases one nonzero entry, preserving the number of nonzeros. Repeating accepted edits produces the updated sparse quantized weights.

Proposition 2 establishes that every accepted pair preserves the target support size and strictly decreases the local surrogate. It does not imply global optimality or monotonic improvement of the true calibration or test loss.

4 REALIGNQ

Sparsification determines which weights remain nonzero, whereas quantization determines how the retained values are represented discretely. Under aggressive compression, support selection and quantized-value assignment can no longer be evaluated independently: whether a coordinate should be retained depends on the quantized values it can realize, while the utility of a quantized value depends on the support context in which it is used. We therefore propose *ReAlignQ*, a post-hoc refinement framework that starts from an already compressed sparse-quantized state rather than constructing a new joint compression pipeline. Using calibration data, ReAlignQ locally repairs support-code mismatch through admissible transitions while preserving the parent model's quantization scales and target support sizes. Figure 1 summarizes the refinement workflow using intuitive terms. Formally, exchange income corresponds to the decrease predicted by the interaction aware local surrogate for a joint support and code transition. The following subsections define quantization aware screening, candidate interactions from per sample gradient signatures, feasible state construction, and the acceptance rule for joint exchanges.

4.1 JOINT SUPPORT AND CODE STATES

Consider a model with L linear weight matrices. ReAlignQ takes an initially compressed model as its reference state, with matrix l written as $W_l^0 = S_l \odot Z_l^0$, where S_l contains fixed quantization scales and Z_l^0 is the integer code matrix. We omit l when discussing a single matrix. The final code q_i jointly encodes whether coordinate i belongs to the nonzero support and, if so, which quantized value it takes: $q_i = 0$ places the coordinate outside the final nonzero support, whereas $q_i \neq 0$ includes it with a specific discrete value. A transition from z_i^0 to q_i induces the physical change $\Delta_i(q_i) = s_i(q_i - z_i^0)$.

For a coordinate that is zero in the reference state, the admissible states are $\{0\} \cup \mathcal{Q}_i^{\mathrm{nz}}$, where $\mathcal{Q}_i^{\mathrm{nz}}$ is a finite set of admissible nonzero codes. For a coordinate that is nonzero in the reference state, they are $\{z_i^0, 0\}$. We denote either set by $\mathcal{Q}_i^{\mathrm{adm}}$. All unedited coordinates and quantization scales remain fixed. For each linear weight matrix l, define the feasible set

$$\mathcal{F}_l = \left\{ q_l : \|q_l\|_0 = K_l, \quad q_{l,i} \in \mathcal{Q}_{l,i}^{\mathrm{adm}} \text{ for all } i \right\}, \tag{5}$$

where K_l is the prescribed support size. The variable $q_{l,i}$ determines both whether coordinate i belongs to the support and which discrete value it takes.

ReAlignQ constructs a feasible state for each linear weight matrix. It uses a local surrogate defined over screened candidates. The surrogate is built from gradients of the full model calibration loss evaluated at the common reference state W^0.

4.2 Screening Activations and Releases

Enumerating the states in $\mathcal{F}_l$ is combinatorial. ReAlignQ first screens possible activations and releases using inexpensive diagonal statistics. It then models interactions among retained candidates in each linear weight matrix. Let $\mathcal{L}_{\text{cal}} = N^{-1}\sum_{t=1}^{N} L_t$, and let $u_{t,i} = \partial L_t(W^0)/\partial w_i$ be the gradient of coordinate i on calibration sample t. These gradients are obtained from full model forward and backward passes at W^0. Their statistics and candidate sets are organized separately for each linear weight matrix. In the first gradient pass, ReAlignQ aggregates

$$g_i = \frac{1}{N}\sum_{t=1}^{N} u_{t,i}, \qquad F_i = \frac{1}{N}\sum_{t=1}^{N} u_{t,i}^2, \tag{6}$$

where g_i gives the first-order direction and F_i is an empirical diagonal Fisher statistic measuring self-sensitivity.

ReAlignQ screens both activation of coordinates that are zero in the reference state and release of coordinates that are nonzero. Activating coordinate i with code q produces the physical change $s_i q$. Releasing coordinate j produces $x_j^{\text{rel}} = -s_j z_j^0$. Their scores are

$$\begin{aligned} C_i^{\text{act}} &= \min_{q \in \mathcal{Q}_i^{\text{nz}}} \left[g_i(s_i q) + \frac{1}{2} F_i (s_i q)^2 \right], \\ C_j^{\text{rel}} &= g_j x_j^{\text{rel}} + \frac{1}{2} F_j \left(x_j^{\text{rel}}\right)^2. \end{aligned} \tag{7}$$

The activation score evaluates support allocation directly through realizable quantized codes rather than an unconstrained continuous value, while the release score measures removal of the reference discrete state. Within each direction, ReAlignQ retains low-score coordinates to form over-complete activation candidates $\mathcal{C}^{\text{zero}}$ and release candidates $\mathcal{C}^{\text{nonzero}}$. Their union $\mathcal{C}$ restricts subsequent evaluation without fixing the final support. The search domain for matrix l is

$$\mathcal{F}_l(\mathcal{C}) = \left\{ q_l \in \mathcal{F}_l : q_{l,i} = z_{l,i}^0 \text{ for all } i \notin \mathcal{C} \right\}. \tag{8}$$

After screening, a second pass over the same calibration data collects gradients from individual samples only for the $M = |\mathcal{C}|$ candidates. This reduces the retained signature storage from $O(NP)$ to $O(NM)$.

The role of the quantization grid in Equation equation 7 is formalized in Proposition 1. For a coordinate that is zero in the reference state, C_i^{act} is the corresponding undamped support increment that accounts for the quantization grid. These scores are intentionally individual and over-complete. Screening reduces the candidate space but does not determine which retained transitions are jointly compatible.

4.3 Local Objective with Candidate Interactions

To model compatibility among candidate transitions in one linear weight matrix, the second gradient pass retains the gradient signature from each sample $U_i = [u_{1,i}, \ldots, u_{N,i}]^\top$ for each $i \in \mathcal{C}$, and stacks the signatures as $U \in \mathbb{R}^{N \times M}$. Thus, g_i, F_i, and U_i come from the same calibration gradient field. Only their retained granularity differs. The inner product $U_i^\top U_j / N$ captures gradient-Gram interaction on the same calibration samples.

For a joint physical change Δ within one linear weight matrix, ReAlignQ uses the following local surrogate:

$$\begin{aligned} J(\Delta) = g^\top \Delta &+ \frac{1}{2}\sum_{i \in \mathcal{C}} F_i \Delta_i^2 \\ &+ \frac{1}{N}\sum_{\substack{i<j \\ i,j \in \mathcal{C}}} U_i^\top U_j \, \Delta_i \Delta_j + \lambda \|\Delta\|_2^2, \end{aligned} \tag{9}$$

where $\lambda \geq 0$ damps large physical changes. The diagonal term counts each self-interaction once because $F_i = \|U_i\|_2^2/N$. The signature term supplies only cross-coordinate interactions. Since these terms act on $\Delta_i = s_i(q_i - z_i^0)$, they directly depend on the scales, codes, and support states of the realized quantized transitions. Each linear weight matrix has its own surrogate. ReAlignQ includes interactions among candidates in the same matrix. It does not construct interaction terms across different matrices. The search compares states in $\mathcal{F}_l(\mathcal{C})$. Theorem 1 quantifies the interaction omitted by diagonal screening and motivates recovering sample-wise interactions within the candidate set. The resulting local surrogate evaluates a joint collection of transitions, but does not by itself specify how to construct a feasible sparse-quantized state under the target support constraint.

4.4 Coupled Support and Code Selection

Evaluation from the current state. ReAlignQ therefore constructs and refines a feasible state using marginals that depend on the current state and exchanges evaluated for each pair. Let Δ be the current candidate-space edit and $r = U\Delta$ its sample-wise response. For an unedited coordinate i, the marginal surrogate change caused by a physical action xe_i is

$$m_i(x \mid \Delta) = \left(g_i + \frac{1}{N}U_i^\top r\right) x + \frac{1}{2}(F_i + 2\lambda)x^2. \tag{10}$$

This marginal changes with the current iterate through its interaction with r. A release candidate j is evaluated as $m_j(x_j^{\mathrm{rel}} \mid \Delta)$. For an activation candidate i, ReAlignQ jointly evaluates entry into the support and the incoming code:

$$q_i^\star(\Delta) \in \arg\min_{q \in \mathcal{Q}_i^{\mathrm{nz}}} m_i(s_i q \mid \Delta). \tag{11}$$

The current iterate therefore changes not only whether a coordinate should be activated, but also which admissible quantized code it should receive. The solution $q_i^\star(\Delta)$ is used for individual activation evaluation and feasible-state construction. During paired exchange, the incoming code is re-optimized for the specific release candidate.

Joint exchange and code selection. Let $\mathcal{A} \subseteq \mathcal{C}^{\mathrm{zero}}$ and $\mathcal{D} \subseteq \mathcal{C}^{\mathrm{nonzero}}$ denote the selected activations and releases, respectively. If $n_0 = \|Z^0\|_0$, feasibility requires $|\mathcal{A}| - |\mathcal{D}| = K - n_0$. Depending on the sign of $K - n_0$, ReAlignQ first activates or releases coordinates using the coupled marginals until the target support size is reached, updating Δ and r after each transition. It then reallocates support through paired activation and release updates, which preserve that support size.

For an activation candidate i, a release candidate j, and an admissible incoming code q, define $x_i(q) = s_i q$ and $x_j^{\mathrm{rel}} = -s_j z_j^0$. Relative to the same current iterate, the complete pair change is

$$\begin{aligned}\Delta J_{i,j}^{\mathrm{pair}}(q) = m_i\big(x_i(q) \mid \Delta\big) + m_j\big(x_j^{\mathrm{rel}} \mid \Delta\big) \\ + \frac{1}{N}U_i^\top U_j\, x_i(q) x_j^{\mathrm{rel}}.\end{aligned} \tag{12}$$

The first two terms account for both actions against the current iterate, while the last adds their direct interaction. Because this interaction depends on the incoming code, ReAlignQ re-optimizes that code for every candidate pair:

$$q_i^\star(j \mid \Delta) \in \arg\min_{q \in \mathcal{Q}_i^{\mathrm{nz}}} \Delta J_{i,j}^{\mathrm{pair}}(q). \tag{13}$$

Thus, the same objective selects the outgoing coordinate, the incoming coordinate, and the discrete value assigned to the incoming coordinate.

Let $\tau \geq 0$ be the acceptance margin. ReAlignQ accepts a pair when

$$\Delta J_{i,j}^{\mathrm{pair}}\big(q_i^\star(j \mid \Delta)\big) < -\tau. \tag{14}$$

The feasibility and descent properties of an accepted pair are stated in Proposition 2. ReAlignQ performs an approximate search in the restricted candidate space rather than enumerating all possible support and code states.

Overall, ReAlignQ uses the same local objective to select which current nonzero is released, which zero coordinate is activated, and which legal code is assigned to the incoming weight. It models interactions among these transitions within each linear weight matrix.

4.5 Sparse and Quantized Operator Support

To support the arbitrary nonzero patterns considered by ReAlignQ, we implement a CUDA backend based on an 8×8 bitmap sparse layout. Each tile uses a 64-bit bitmap to encode arbitrary retained positions and stores their quantized values compactly in bitmap order, without imposing an $N{:}M$ constraint. The backend provides W4A16, W8A16, and W8A8 execution paths. For W8A8, integer weights and activations are accumulated with $\mathrm{s8} \times \mathrm{s8} \rightarrow \mathrm{s32}$ Tensor Core operations, followed by scale application in the output stage. Its exact bitmap path retains a nonuniform input mask without uniformization through a padded layout with offsets. A separate uniform layout is used only when every tile already satisfies that stronger layout constraint. The metadata formats, scale semantics, CUDA data flow, activation quantization, correctness conditions, and implementation limitations are detailed in Appendix B.

5 Experiments

We evaluate whether ReAlignQ improves compressed parent models under fixed scales and sparsity, transfers across compression regimes, model families, and parent methods, and benefits from each of its components. Every comparison is paired between a compressed reference model and its refined counterpart under the same quantization scales, target sparsity, and evaluation protocol.

5.1 Experimental Setup

Models and evaluation tasks. We evaluate Llama 3 (Llama Team, AI at Meta, 2024) and Qwen3 (Qwen Team, 2025) models, covering both model-family and model-scale variation. Language-modeling quality is measured by WikiText-2 perplexity (Merity et al., 2017). We evaluate zero-shot transfer on PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), and WinoGrande (Sakaguchi et al., 2020), reporting each task and their unweighted mean. WikiText-2 uses 32 non-overlapping windows of length 2,048, yielding 65,504 predictions from a fixed 65,536-token prefix.

Calibration and compression settings. Compression and refinement use fixed C4 samples (Raffel et al., 2020). Our primary setting is unstructured weight sparsity with W8A16 quantization. We additionally evaluate W8A8 and use W4A16 as a more aggressive quantization stress test. ReAlignQ modifies only admissible support and code states while preserving each parent's quantization scales and target support size.

Baselines and reference models. We refine parents produced by JSQ (Guo et al., 2024), OBR-GPTQ (Guo et al., 2026), EIWR (Xie et al., 2026), and SparseGPT followed by GPTQ (Frantar & Alistarh, 2023; Frantar et al., 2023). These methods span joint compression, post-compression restoration, post-pruning refinement, and sequential sparsification and quantization. Each parent is compared directly with its ReAlignQ-refined counterpart, isolating the effect of refinement from differences between parent pipelines.

Implementation and reporting protocol. Unless stated otherwise, candidate budgets, damping, stopping criteria, and evaluators are fixed across models. All main-table entries use the same WikiText-2 and zero-shot protocols. Appendix 7 summarizes the solver settings.

5.2 Main Results under Joint Compression

Table 1 compares ReAlignQ with strong compression baselines across model families and sizes.

Overall comparison. ReAlignQ lowers PPL for every model and compression method in Table 1. With EIWR, it reduces PPL from 35.48 to 25.57 on Llama3-8B, from 24.81 to 20.81 on Qwen3-8B, and from 19.43 to 16.24 on Qwen3-14B; the Llama3-8B zero-shot average also improves by 2.6 points. Gains across JSQ, OBR-GPTQ, EIWR, and SparseGPT+GPTQ support its use as a post-hoc refinement across compression pipelines.

Table 1: Main results on Llama and Qwen models under W8A16 quantization and 70% unstructured sparsity. Lower WikiText-2 perplexity (PPL) and higher zero-shot accuracy are better. Each "+ ReAlignQ" row refines the corresponding compressed model. Parenthetical values report refined-minus-parent changes in PPL and zero-shot average (percentage points), respectively.

	Language Modeling	Zero-Shot Accuracy (%)						
Method	Wiki2 PPL↓	PIQA	BoolQ	Hella.	ARC-e	ARC-c	Wino.	Avg.↑
Llama 3 8B								
JSQ	133.80	55.1	38.3	30.2	32.0	19.7	50.8	37.7
+ ReAlignQ	119.23 (-14.57)	55.9	38.3	30.2	32.2	20.5	51.7	38.1 (+0.4)
OBR-GPTQ	101.82	56.4	61.1	30.7	32.1	21.3	49.6	41.9
+ ReAlignQ	93.42 (-8.40)	57.3	62.8	31.8	31.9	21.5	49.9	42.5 (+0.6)
EIWR	35.48	65.1	68.9	43.3	43.5	25.9	58.2	50.8
+ ReAlignQ	**25.57** (-9.91)	**67.1**	68.7	**49.0**	**47.3**	**27.5**	**61.0**	**53.4** (+2.6)
SparseGPT+GPTQ	53.09	60.2	63.4	42.4	50.6	22.4	50.1	48.2
+ ReAlignQ	32.93 (-20.16)	65.4	64.5	43.4	57.8	25.6	53.1	51.6 (+3.4)
Qwen3 8B								
JSQ	69.00	59.8	61.8	34.3	42.9	22.9	52.1	45.6
+ ReAlignQ	65.02 (-3.98)	60.6	62.9	34.7	43.0	23.3	51.7	46.0 (+0.4)
OBR-GPTQ	59.99	60.4	62.3	35.1	37.4	24.1	53.8	45.5
+ ReAlignQ	55.16 (-4.83)	62.5	62.4	36.1	39.3	24.1	55.2	46.6 (+1.1)
EIWR	24.81	68.6	72.0	51.0	57.1	34.4	61.2	57.4
+ ReAlignQ	**20.81** (-4.00)	70.3	72.5	**54.7**	**58.4**	**35.6**	**62.8**	**59.1** (+1.7)
SparseGPT+GPTQ	24.85	70.0	78.4	49.3	56.8	33.7	62.7	58.5
+ ReAlignQ	23.36 (-1.49)	**70.4**	**78.6**	50.5	57.6	34.1	63.5	**59.1** (+0.6)
Qwen3 14B								
JSQ	97.95	66.3	62.0	41.7	54.4	29.2	57.0	51.8
+ ReAlignQ	82.20 (-15.75)	66.2	63.0	44.7	54.3	30.7	58.2	52.9 (+1.1)
OBR-GPTQ	42.74	65.6	64.3	44.2	51.1	28.8	58.4	52.1
+ ReAlignQ	38.82 (-3.92)	66.6	64.3	46.1	51.8	29.9	59.4	53.0 (+0.9)
EIWR	19.43	73.5	82.1	58.4	**69.4**	**44.7**	68.2	66.0
+ ReAlignQ	**16.24** (-3.19)	73.4	80.3	**62.2**	68.3	44.4	**68.7**	**66.2** (+0.2)
SparseGPT+GPTQ	18.13	72.7	**82.9**	58.4	65.7	39.8	67.9	64.6
+ ReAlignQ	17.25 (-0.88)	**73.7**	82.7	59.2	66.2	40.6	67.4	65.0 (+0.4)

Table 2: Robustness across sparsity and numerical precision on Llama3-8B. Lower PPL and higher zero-shot average are better.

		WikiText-2 PPL↓		Zero-Shot Avg.↑	
W/A	Sparsity	Compressed	ReAlignQ	Compressed	ReAlignQ
8/16	50%	9.94	9.69	67.4	69.5
8/16	60%	15.89	13.56	61.8	63.9
8/16	70%	53.09	32.93	48.2	51.6
8/8	50%	9.90	9.61	66.8	68.7
8/8	60%	16.46	14.56	60.3	62.9
8/8	70%	55.27	34.30	49.2	52.6
4/16	50%	12.33	11.69	65.3	65.3
4/16	60%	24.73	22.81	58.4	60.4
4/16	70%	86.86	71.69	44.8	48.7

5.3 Robustness across Compression Regimes

We evaluate ReAlignQ on Llama3-8B across three sparsity levels and three weight and activation precision settings. This experiment examines whether the observed improvements persist as compression becomes more aggressive.

Sparsity and precision. ReAlignQ lowers PPL in all nine settings in Table 2, with the largest W8 gains at 70% sparsity. It improves zero-shot averages in every W8 setting and in W4A16 at

Table 3: (a) Joint support and code decision and (b) disjoint C4 holdout evaluation on Llama3-8B with EIWR W8A16. Panel (a) reports 60% and 70% sparsity. Panel (b) reports 70%. Lower PPL is better.

(a) Joint support and code decision

Variant	60% PPL↓	70% PPL↓
Support before code	15.01	30.74
Full ReAlignQ	**13.41**	**25.57**

(b) Disjoint C4 holdout

System	Wiki2 PPL↓	C4 PPL↓
EIWR baseline	35.480	30.576
Full ReAlignQ	25.568	23.260

Table 4: Component ablations on Llama3-8B with EIWR W8A16. Lower WikiText-2 PPL is better.

Variant	60% PPL↓	70% PPL↓	Ablated component
Full ReAlignQ	**13.41**	**25.57**	–
continuous-score screening	14.31	27.15	candidate screening over legal codes
diagonal surrogate	14.45	30.56	cross-coordinate interaction
fixed activation-only code	13.84	27.52	incoming code selected for each pair
fill-to-target only	14.32	27.46	bidirectional support reallocation

Table 5: End-to-end deployment evaluation on Llama3-8B under W8A16 quantization and 70% sparsity. Packed PPL is evaluated with the sparse inference operator. Static storage reports the serialized model footprint.

System	HF PPL↓	Packed PPL↓	Prefill tok/s↑	Static Storage (GiB)↓
Dense FP16	6.32	–	8406.0	14.96
EIWR baseline	35.48	35.62	8642.4	3.27
EIWR + ReAlignQ	25.57	25.67	8638.6	3.27

60% and 70% sparsity. PPL also remains within 25.56–25.90 across 32–256 calibration sequences (Appendix 12).

5.4 Ablation Study

Table 4 examines the four internal components of ReAlignQ, while Table 3(a) compares joint support and code selection with a sequential approach that selects the support before assigning codes.

Joint support and code decision. The decoupled baseline first selects support using a code-oblivious continuous criterion and then assigns codes independently. Full ReAlignQ lowers PPL from 15.01 to 13.41 at 60% sparsity and from 30.74 to 25.57 at 70%, with a larger advantage under the more aggressive setting.

Component analysis. As shown in Table 4, removing any component degrades PPL at both sparsity levels, with the diagonal surrogate producing the largest gap at 70%. Pair-level validation shows moderate positive agreement with held-out loss improvement (Spearman 0.36, 66% sign accuracy; Appendix C.5), while the disjoint C4 holdout in Table 3(b) improves from 30.576 to 23.260 PPL.

5.5 Deployment Evaluation

We evaluate the packed deployment on Llama3-8B and compare its HuggingFace evaluation with the sparse inference operator. We report prefill throughput and static storage under the same sequence-length protocol. Packed and HuggingFace PPL differ by at most 0.14. ReAlignQ leaves static storage unchanged at 3.27 GiB and achieves nearly identical prefill throughput to its parent (8638.6 versus 8642.4 tok/s); W8A8 results show the same pattern (Appendix 11).

5.6 Scope and Cost

ReAlignQ is an offline calibration-time refinement stage. On Llama3-8B EIWR W8A16 at 70% sparsity, it uses 3.97 GPU-hours and 37,152 MiB peak GPU memory (Appendix 8). Across model families, compression methods, numerical precisions, and sparsity levels, ReAlignQ improves the evaluated compressed models without changing their deployment footprint. It operates within a restricted candidate space and does not imply global optimality over support and code states.

AI USE STATEMENT

Generative AI tools were used solely for language editing, including improving grammar, clarity, and readability. They were not used to develop research ideas, design the methodology or experiments, analyze data, interpret results, or formulate scientific claims. All AI-assisted edits were reviewed by the authors. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies post-training compression of large language models and does not involve human-subject experiments or the collection of new personal data. The evaluated models and datasets may inherit biases, harmful content, or privacy risks from their original training data. ReAlignQ does not mitigate these underlying risks, and compressed models should therefore be evaluated for safety and fairness before deployment. While model compression may reduce storage and inference costs, its broader environmental and societal effects depend on the deployment setting and usage scale.

REFERENCES

Saleh Ashkboos, Maximilian Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/316648eb8b4ffb6010f531b07848c300-Abstract-Conference.html.

Saleh Ashkboos, Ilia Markov, Elias Frantar, Tingxuan Zhong, Xincheng Wang, Jie Ren, Torsten Hoefler, and Dan Alistarh. QUIK: Towards end-to-end 4-bit inference on generative large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3355–3371. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.emnlp-main.197. URL https://aclanthology.org/2024.emnlp-main.197/.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2924–2936. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

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. doi: 10.48550/arXiv.1803.05457. URL https://arxiv.org/abs/1803.05457.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Q1u25ahSuy.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12284–12303. PMLR, 2024. URL https://proceedings.mlr.press/v235/egiazarian24a.html.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4475–4488. Curran Associates, Inc., 2022. doi: 10.52202/068431-0323. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/1caf09c9f4e6b0150b06a07e77f2710c-Abstract.html.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 2023. URL https://proceedings.mlr.press/v202/frantar23a.html.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://iclr.cc/virtual/2023/poster/10855.

Hang Guo, Luca Benini, and Yawei Li. Optimal brain restoration for joint quantization and sparsification of LLMs. In *International Conference on Learning Representations*, volume 2026, pp. 140852–140869, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/e45d826ee52dce029198b6968a10423e-Abstract-Conference.html.

Jinyang Guo, Jianyu Wu, Zining Wang, Jiaheng Liu, Ge Yang, Yifu Ding, Ruihao Gong, Haotong Qin, and Xianglong Liu. Compressing large language models by joint sparsification and quantization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16945–16957. PMLR, 2024. URL https://proceedings.mlr.press/v235/guo24g.html.

Simla Burcu Harma, Ayan Chakraborty, Elizaveta Kostenok, Danila Mishin, Dongho Ha, Babak Falsafi, Martin Jaggi, Ming Liu, Yunho Oh, Suvinay Subramanian, and Amir Yazdanbakhsh. Effective interplay between sparsity and quantization: From theory to practice. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ed032b08a8822c3635cdcd961012ce60-Abstract-Conference.html.

Jinuk Kim, Marwa El Halabi, Wonpyo Park, Clemens Js Schaefer, Deokjae Lee, Yeonhong Park, Jae W. Lee, and Hyun Oh Song. GuidedQuant: Large language model quantization via exploiting end loss guidance. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 30011–30037. PMLR, 2025. URL https://proceedings.mlr.press/v267/kim25d.html.

Sehoon Kim, Coleman Richard Charles Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 23901–23923. PMLR, 2024. URL https://proceedings.mlr.press/v235/kim24f.html.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 quantization and system co-design for efficient LLM serving. In *Proceedings of Machine Learning and Systems*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/file/fbe2b2f74a2ece8070d8fb073717bda6-Paper-Conference.pdf.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ogO6DGE6FZ.

Llama Team, AI at Meta. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783. URL https://arxiv.org/abs/2407.21783.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Mohammad Mozaffari, Amir Yazdanbakhsh, and Maryam Mehri Dehnavi. SLiM: One-shot quantization and sparsity with low-rank approximation for LLM weight compression. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45024–45049. PMLR, 2025. URL https://proceedings.mlr.press/v267/mozaffari25a.html.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

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.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8Wuvhh0LYW.

Oliver Sieberling, Denis Kuznedelev, Eldar Kurtic, and Dan Alistarh. EvoPress: Accurate dynamic model compression via evolutionary search. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55556–55590. PMLR, 2025. URL https://proceedings.mlr.press/v267/sieberling25a.html.

Yupeng Su, Ziyi Guan, Xiaoqun Liu, Tianlai Jin, Dongkuan Wu, Zhengfei Chen, Graziano Chesi, Ngai Wong, and Hao Yu. LLM-Barber: Block-aware rebuilder for sparsity mask in one-shot for large language models. In *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9. IEEE, 2025. doi: 10.1109/ICCAD66269.2025.11240704. URL https://doi.org/10.1109/ICCAD66269.2025.11240704.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/14c856c7a41297804de4c4890e846b25-Abstract-Conference.html.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. FlatQuant: Flatness matters for LLM quantization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57587–57613. PMLR, 2025. URL https://proceedings.mlr.press/v267/sun25l.html.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

Jingcheng Xie, Yinda Chen, Xiaoyu Liu, Yinglong Li, Haoyuan Shi, and Zhiwei Xiong. Efficient plug-and-play weight refinement for sparse large models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27081–27089, 2026. doi: 10.1609/aaai.v40i32.39922. URL https://ojs.aaai.org/index.php/AAAI/article/view/39922.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Chao Zeng, Songwei Liu, Shu Yang, Fangmin Chen, Xing Mei, and Lean Fu. GQSA: Group quantization and sparsity for accelerating large language model inference. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 149–165. The Asian Federation of Natural Language Processing and the Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.ijcnlp-long.10. URL https://aclanthology.org/2025.ijcnlp-long.10/.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. Plug-and-play: An efficient post-training pruning method for large language models. In *International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=Tr0lPx9woF.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Yunyun Sun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. Dynamic sparse no training: Training-free fine-tuning for sparse LLMs. In *International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=1ndDmZdT4g.

A Detailed Theory of Support-Code Transitions

This appendix proves the three results stated in Section 3 and gives an auxiliary fidelity analysis for the local surrogate. Unless stated otherwise, all arguments concern one linear layer and its restricted candidate space. Coordinates outside the candidate set remain fixed, every candidate scale is fixed and strictly positive ($s_i > 0$), and every admissible code set is finite. Thus, an admissible nonzero code represents a physically nonzero quantized weight; this equivalence would fail for a zero scale. These restrictions match the optimization domain of ReAlignQ; none of the results below imply unrestricted full-model optimality, guaranteed calibration-loss improvement, or improved test performance. We use the support-code notation of Section 4; in particular, $\Delta_i(q_i) = s_i(q_i - z_i^0)$ denotes a physical candidate-space transition from the reference state.

The role of this analysis is explanatory rather than a claim of full-model convergence. Specifically, the results formalize why code-aware support scoring, candidate-level interaction modeling, and support-preserving exchanges are appropriate design choices for the restricted search performed by ReAlignQ. The correspondence between the resulting local surrogate and calibration or downstream performance is evaluated empirically.

A.1 Quantization-Aware Support Selection

Candidate-space cardinality. Let $\mathcal{C} = \{1, \dots, M\}$ be the candidate set, and let

$$K_{\text{fixed}} = \left\| z^0_{\mathcal{C}^c} \right\|_0$$

be the number of nonzeros fixed outside it. For a layer target K, the candidate-space target used in Proposition 1 is

$$K_{\mathcal{C}} = K - K_{\text{fixed}}.$$

We assume that at least one admissible candidate state has exactly $K_{\mathcal{C}}$ nonzero codes. In this subsection, q denotes the candidate-space code vector indexed only by $\mathcal{C}$; the full-layer state keeps $z^0_{\mathcal{C}^c}$ fixed. Thus, $\|q\|_0 = K_{\mathcal{C}}$ is equivalent to the full-layer cardinality constraint $\|z\|_0 = K$.

General support reduction. For a support $S \subseteq \mathcal{C}$, define

$$\mathcal{Q}(S) = \left\{ q : q_i \in \mathcal{Q}_i^{\text{adm}},\ \text{supp}(q) = S \right\}.$$

For any objective $\Phi(q)$ over the finite admissible codebook, let

$$V(S) = \begin{cases} \min\limits_{q \in \mathcal{Q}(S)} \Phi(q), & \mathcal{Q}(S) \neq \varnothing, \\ +\infty, & \mathcal{Q}(S) = \varnothing. \end{cases}$$

The feasible code vectors are partitioned by their supports, and hence

$$\min_{\substack{q_i \in \mathcal{Q}_i^{\text{adm}} \\ \|q\|_0 = K_{\mathcal{C}}}} \Phi(q) = \min_{\substack{S \subseteq \mathcal{C} \\ |S| = K_{\mathcal{C}}}} V(S). \tag{15}$$

Thus, a support can be scored consistently with the joint problem only after optimizing the codes realizable on that support.

Proof of Proposition 1. The exact diagonal solution additionally assumes that every candidate coordinate admits code zero and at least one admissible nonzero code, that $0 \le K_{\mathcal{C}} \le M$, and that $C = \text{Diag}(F_1, \dots, F_M)$. Ties among the $K_{\mathcal{C}}$-smallest values of δ_i may be resolved arbitrarily. These assumptions make the diagonal candidate-space objective separable and ensure that every unselected coordinate can take code zero.

For each candidate coordinate, define

$$\phi_i(q) = g_i \Delta_i(q) + \frac{1}{2}(F_i + 2\lambda)\Delta_i(q)^2, \qquad \delta_i = \min_{q \in \mathcal{Q}_i^{\text{adm}} \setminus \{0\}} \phi_i(q) - \phi_i(0).$$

Under the diagonal assumption, the candidate-space objective is separable:

$$\Phi_{\text{diag}}(q) = \sum_{i=1}^{M} \phi_i(q_i),$$

For any feasible support S of size $K_{\mathcal{C}}$, its best admissible code assignment has value

$$V(S) = \sum_{i \notin S} \phi_i(0) + \sum_{i \in S} \min_{q \in \mathcal{Q}_i^{\mathrm{adm}} \setminus \{0\}} \phi_i(q)$$
$$= \sum_{i=1}^{M} \phi_i(0) + \sum_{i \in S} \delta_i. \tag{16}$$

The first term in Equation equation 16 is independent of S. Minimizing $V(S)$ over supports of size $K_{\mathcal{C}}$ therefore amounts to selecting the $K_{\mathcal{C}}$ smallest values of δ_i. For every selected coordinate, finiteness of the admissible code set guarantees that a minimizing nonzero code is attained. All unselected coordinates take code zero, proving the proposition. □

Relation to the screening score. If coordinate i is zero in the reference state, then $\Delta_i(0) = 0$ and $\phi_i(0) = 0$. With the undamped screening convention $\lambda = 0$,

$$\delta_i = \min_{q \in \mathcal{Q}_i^{\mathrm{nz}}} \left[g_i(s_i q) + \frac{1}{2} F_i (s_i q)^2 \right] = C_i^{\mathrm{act}}.$$

Consequently, δ_i is the theoretical support increment represented by the existing activation score, not a third score used by the algorithm.

Continuous targets and coupled support values. For a reference-zero coordinate, relaxing the admissible codebook to the real line can yield a continuous target, but that target is only a search aid: the implemented transition must minimize over the finite admissible code set. For an interval codebook, the legal minimizer lies at a nearest feasible code or a boundary; for an irregular codebook, the legal states must be enumerated. Once off-diagonal interactions are present, the value of a support also depends on the joint signs and magnitudes of its codes. Consequently, no fixed per-coordinate score need rank all supports correctly in the coupled problem.

Why continuous support-first selection can fail. Consider

$$\mathcal{L}(w) = \frac{1}{2} \left[(w_1 - 0.6)^2 + (w_2 - 0.5)^2 \right], \qquad w^0 = (0, 0), \qquad \|w\|_0 = 1. \tag{17}$$

In continuous space, selecting coordinate 1 achieves loss 0.125, whereas selecting coordinate 2 achieves 0.18; an exact continuous support optimizer therefore selects coordinate 1.

Now let both integer codebooks be $\{-1, 0, 1\}$, but use scales $s_1 = 2$ and $s_2 = 0.5$. The best nonzero state on coordinate 1 is $w_1 = 2$, giving

$$\mathcal{L}(2, 0) = \frac{1}{2}(1.4^2 + 0.5^2) = 1.105.$$

Coordinate 2 can realize $w_2 = 0.5$ exactly, giving $\mathcal{L}(0, 0.5) = 0.18$. The joint legal solution therefore selects coordinate 2, opposite to continuous support-first selection. Because Equation equation 17 is itself quadratic, this disagreement is caused entirely by the quantization grid rather than Taylor approximation.

The gap can be unbounded. Replacing $s_1 = 2$ with $s_1 = R$ gives

$$\mathcal{L}_{\mathrm{SF}}(R) = \frac{1}{2} \left[(R - 0.6)^2 + 0.25 \right],$$

while the legal state on coordinate 2 retains loss 0.18. For sufficiently large R, it remains the joint legal optimum and both the additive gap and the ratio $\mathcal{L}_{\mathrm{SF}}(R)/0.18$ diverge.

A.2 Interaction Bounds for Gradient Signatures

Let

$$G = \frac{1}{N} U^\top U, \qquad F_i = G_{ii} = \frac{1}{N} \|U_i\|_2^2.$$

The results in this subsection assume this diagonal identity. Define the active transition set $S = \mathrm{supp}(\Delta)$, with $|S| = k$, and

$$D(\Delta) = \frac{1}{2}\sum_{i\in S} F_i\Delta_i^2, \qquad I(\Delta) = \frac{1}{N}\sum_{\substack{i<j\\ i,j\in S}} U_i^\top U_j\,\Delta_i\Delta_j.$$

Define the nonzero-signature index set

$$\mathcal{I}_+ = \{i : \|U_i\|_2 > 0\}.$$

When $|\mathcal{I}_+| \geq 2$, the signature coherence is

$$\mu = \max_{\substack{i\neq j\\ i,j\in\mathcal{I}_+}} \frac{|U_i^\top U_j|}{\|U_i\|_2\|U_j\|_2};$$

when $|\mathcal{I}_+| \leq 1$, we set $\mu = 0$. By Cauchy–Schwarz, $0 \leq \mu \leq 1$.

Proof of Theorem 1. Let $S = \mathrm{supp}(\Delta)$ with $|S| = k$, and define

$$a_i = \frac{\|U_i\|_2}{\sqrt{N}}|\Delta_i| = \sqrt{F_i}|\Delta_i|.$$

The coherence definition gives

$$\begin{aligned}
|I(\Delta)| &\leq \frac{\mu}{N}\sum_{\substack{i<j\\ i,j\in S}} \|U_i\|_2\|U_j\|_2|\Delta_i||\Delta_j| \\
&= \mu\sum_{\substack{i<j\\ i,j\in S}} a_i a_j \\
&= \frac{\mu}{2}\left[\left(\sum_{i\in S} a_i\right)^2 - \sum_{i\in S} a_i^2\right]. \qquad (18)
\end{aligned}$$

Applying

$$\left(\sum_{i\in S} a_i\right)^2 \leq k\sum_{i\in S} a_i^2$$

yields

$$|I(\Delta)| \leq \frac{\mu(k-1)}{2}\sum_{i\in S} F_i\Delta_i^2 = \mu(k-1)D(\Delta),$$

which proves the theorem. □

Curvature sandwich. The candidate Gram curvature decomposes as

$$Q_{\mathrm{cpl}}(\Delta) = \frac{1}{2N}\|U\Delta\|_2^2 = D(\Delta) + I(\Delta).$$

Theorem 1 and $Q_{\mathrm{cpl}}(\Delta) \geq 0$ imply

$$\max\{0, 1-\mu(k-1)\}D(\Delta) \leq Q_{\mathrm{cpl}}(\Delta) \leq [1+\mu(k-1)]D(\Delta). \qquad (19)$$

If $\mu(k-1) < 1$, diagonal and coupled curvature control each other on the fixed k-sparse transition. If $\mu(k-1) \geq 1$, the worst-case lower bound becomes uninformative. Equation equation 19 concerns only curvature; it does not provide the same multiplicative guarantee for the complete surrogate, whose first-order term may have either sign.

The factor $\mu(k-1)$ is tight over signature families with no additional rank restriction, in particular when $N \geq k$. Equality is attained when all active a_i are equal and all normalized signature inner products have magnitude μ with signs aligned to the transition. For $N \geq k$, such signatures can be realized by the positive-semidefinite Gram matrix

$$R = (1-\mu)I + \mu\mathbf{1}\mathbf{1}^\top.$$

For a fixed $N < k$, the rank constraint on the Gram matrix may permit a strictly sharper specialized bound; no tightness claim is made for that regime.

Equal diagonal terms need not imply equal coupled cost. Let $N = 2$ and take

$$U_1 = U_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad U_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

All three coordinates have $F_i = 1/2$. For unit physical transitions, the diagonal curvature of any two-coordinate choice is one half. However,

$$Q_{\mathrm{cpl}}(e_1 + e_2) = 1, \qquad Q_{\mathrm{cpl}}(e_1 + e_3) = \frac{1}{2}.$$

Thus, even exact knowledge of all diagonal terms cannot recover the joint cost when signature geometry differs. The coherence result controls the error for a fixed transition; it does not show that a diagonal shortlist contains the coupled optimum. Candidate truncation can therefore introduce approximation bias even if optimization within the retained pool is exact.

Interaction diagnostics. The global coherence μ can be dominated by a small number of pairs. Accordingly, its empirical interpretation should include quantiles and signs of normalized signature similarities within reference-zero/reference-zero, reference-nonzero/reference-nonzero, and cross-group pairs. For a selected transition, the data-dependent ratio

$$R_{\mathrm{int}} = \begin{cases} |I(\Delta)|/D(\Delta), & D(\Delta) > 0, \\ 0, & D(\Delta) = 0 \end{cases}$$

is often more directly informative than the worst-case factor $\mu(k-1)$. Changes in candidate ranking or code choice after adding the signature terms should likewise be reported as empirical diagnostics, not deduced from the coherence bound alone.

Marginal interaction bound. For a current transition supported on S, define

$$r = U\Delta, \qquad c_i(\Delta) = \frac{1}{N} U_i^\top r$$

for a coordinate $i \notin S$. Coherence and Cauchy–Schwarz yield

$$|c_i(\Delta)| \le \mu\sqrt{F_i} \sum_{j \in S} \sqrt{F_j}|\Delta_j| \le \mu\sqrt{2kF_i D(\Delta)}. \tag{20}$$

For an activation code q, the interaction contribution is therefore bounded by

$$|c_i(\Delta) s_i q| \le |s_i q|\, \mu\sqrt{2kF_i D(\Delta)}.$$

This expression makes explicit why the scale, code magnitude, active-set size, and signature coherence all affect the reliability of an independent diagonal score.

Stability of a diagonal code choice. Define the diagonal activation cost

$$d_i(q) = g_i s_i q + \frac{1}{2}(F_i + 2\lambda) s_i^2 q^2,$$

and let

$$q_0 \in \arg\min_{q \in \mathcal{Q}_i^{\mathrm{nz}}} d_i(q).$$

After accounting for the current interaction, the cost becomes $\widetilde{d}_i(q) = d_i(q) + c_i(\Delta) s_i q$. If

$$d_i(q) - d_i(q_0) > |c_i(\Delta) s_i (q - q_0)|, \qquad \forall q \ne q_0, \tag{21}$$

then $\widetilde{d}_i(q) > \widetilde{d}_i(q_0)$ for every $q \ne q_0$, so the diagonal code remains optimal for the coupled marginal. Combining Equations equation 20 and equation 21 gives the sufficient condition

$$d_i(q) - d_i(q_0) > \mu |s_i (q - q_0)| \sqrt{2kF_i D(\Delta)}.$$

Uncentered Gram and diagonal mismatch. The signatures define an uncentered Gram matrix. If $U_c = U - \mathbf{1}g^\top$, then

$$\frac{1}{N}U^\top U = \frac{1}{N}U_c^\top U_c + gg^\top.$$

Thus, gradient-Gram interaction contains both centered sample-wise covariance and a mean-gradient component. It should not be described as centered correlation.

If a separately collected diagonal $\widetilde{F}_i$ differs from G_{ii}, the associated curvature is

$$C = G + \mathrm{Diag}\Big(\widetilde{F} - \mathrm{diag}(G)\Big),$$

and the extra quadratic discrepancy satisfies

$$\left|\frac{1}{2}\sum_i (\widetilde{F}_i - G_{ii})\Delta_i^2\right| \le \frac{1}{2}\max_i|\widetilde{F}_i - G_{ii}|\|\Delta\|_2^2.$$

Moreover,

$$\|H - C\|_{\mathrm{op}} \le \|H - G\|_{\mathrm{op}} + \max_i|\widetilde{F}_i - G_{ii}|.$$

This separates mismatch between the true Hessian and the empirical gradient Gram from error due to collecting the diagonal separately. Unlike G, the corrected matrix C need not be positive semidefinite. A sufficient condition for it to be positive semidefinite is $\widetilde{F}_i \ge G_{ii}$ for all i; more generally, the effective curvature $C + 2\lambda I$ is positive semidefinite whenever $\lambda \ge -\lambda_{\min}(C)/2$.

A.3 Exchange Feasibility and Descent

Candidate-space feasibility. In this subsection, q denotes the full-layer code vector: for $i \notin \mathcal{C}$, $q_i = z_i^0$ remains fixed, and for $i \in \mathcal{C}$, $q_i \in \mathcal{Q}_i^{\mathrm{adm}}$. Hence $\|q\|_0 = K$ is the layer-level cardinality constraint and is equivalent to $\|q_{\mathcal{C}}\|_0 = K_{\mathcal{C}}$. Let $\mathcal{C}^{\mathrm{zero}}$ be the reference-zero activation candidate set and $\mathcal{C}^{\mathrm{nonzero}}$ the reference-nonzero release candidate set, with $h = |\mathcal{C}^{\mathrm{zero}}|$ and $s = |\mathcal{C}^{\mathrm{nonzero}}|$. Let $\mathcal{A} \subseteq \mathcal{C}^{\mathrm{zero}}$ and $\mathcal{D} \subseteq \mathcal{C}^{\mathrm{nonzero}}$ denote selected activations and releases. Assume that every coordinate in $\mathcal{C}^{\mathrm{zero}}$ admits at least one physically nonzero code and that every coordinate in $\mathcal{C}^{\mathrm{nonzero}}$ can be released to zero. If $n_0 = \|Z^0\|_0$, then

$$\|q\|_0 = n_0 + |\mathcal{A}| - |\mathcal{D}|. \tag{22}$$

Indeed, every activation changes a reference zero to a nonzero code, every release changes a reference nonzero to zero, and all other support states are fixed. Hence a target K is feasible within the candidate pools if and only if

$$-s \le K - n_0 \le h, \tag{23}$$

or equivalently $n_0 - s \le K \le n_0 + h$.

Exact pair decomposition. Let the local surrogate have quadratic curvature $C = C^\top$, where in ReAlignQ $C_{ij} = U_i^\top U_j/N$ for $i \ne j$. Consider physical actions $a_i e_i$ and $a_j e_j$ that are both absent from the current edit Δ. Their separate marginals are

$$m_i(a_i \mid \Delta) = J(\Delta + a_i e_i) - J(\Delta), \qquad m_j(a_j \mid \Delta) = J(\Delta + a_j e_j) - J(\Delta).$$

Expanding the quadratic gives

$$\begin{aligned} &J(\Delta + a_i e_i + a_j e_j) - J(\Delta) \\ &\quad = m_i(a_i \mid \Delta) + m_j(a_j \mid \Delta) + C_{ij}a_i a_j. \end{aligned} \tag{24}$$

The two marginals contain the current-state linear terms, both self-quadratic terms, and damping. The only missing term is the direct interaction

$$\frac{1}{2}\left(C_{ij}a_i a_j + C_{ji}a_j a_i\right) = C_{ij}a_i a_j.$$

Equation equation 24 is exactly Equation equation 12 for $a_i = x_i(q)$, $a_j = x_j^{\mathrm{rel}}$, and $C_{ij} = U_i^\top U_j/N$.

For a fixed release candidate j, the pair-conditioned incoming code is

$$q_i^\star(j \mid \Delta) \in \arg\min_{q \in \mathcal{Q}_i^{\mathrm{nz}}} \left[m_i(s_i q \mid \Delta) + C_{ij} s_i q \, a_j \right].$$

If a procedure preselects q using only $m_i(s_i q \mid \Delta)$, it uses an independent-code approximation rather than pairwise joint code minimization. Equation equation 24 remains exact for the particular fixed code it evaluates, but not for an unperformed pair-conditioned minimization.

If the second marginal is instead evaluated after applying the first action, then

$$J(\Delta + a_i e_i + a_j e_j) - J(\Delta) = m_i(a_i \mid \Delta) + m_j(a_j \mid \Delta + a_i e_i),$$

and the direct interaction is already included in the second marginal; it must not be added again.

Proof of Proposition 2. Fix $\tau \geq 0$, and assume $\|q^{(t)}\|_0 = K$. The paired update acts on two coordinates not yet changed in the current edit: it sets one coordinate that is currently zero in $q^{(t)}$ to a physically nonzero admissible code and one coordinate that is currently nonzero in $q^{(t)}$ to zero. These changes add and remove exactly one support element, respectively, so

$$\|q^{(t+1)}\|_0 = K + 1 - 1 = K.$$

Because Equation equation 12 is the exact marginal of the fixed quadratic surrogate for the pair, Equation equation 14 directly gives

$$J\big(\Delta^{(t+1)}\big) - J\big(\Delta^{(t)}\big) < -\tau.$$

This proves both conclusions. □

For T accepted exchanges, summing the per-step inequalities gives

$$J\big(\Delta^{(T)}\big) < J\big(\Delta^{(0)}\big) - T\tau \quad \text{for } T \geq 1. \tag{25}$$

For $T = 0$, the two objective values are equal.

Finite termination. The feasible candidate-state set

$$\mathcal{F}_K = \big\{ q : q_i \in \mathcal{Q}_i^{\mathrm{adm}},\ \|q\|_0 = K \big\}$$

is finite. An uncapped search that accepts only strict surrogate descent cannot revisit a previous state, so it accepts at most $|\mathcal{F}_K| - 1$ exchanges, provided every iteration scans a finite set of legal actions and stops when no acceptable action remains. A fixed iteration cap independently ensures termination, but does not imply that the final state is locally optimal. For forward-only add/delete exchanges from $(\mathcal{A}_0, \mathcal{D}_0)$, the combinatorial bound is

$$T_{\mathrm{fwd}} \leq \min\{h - |\mathcal{A}_0|,\ s - |\mathcal{D}_0|\}.$$

If every accepted exchange decreases the surrogate by more than $\tau > 0$, then the finite lower value $J_{\min} = \min_{q \in \mathcal{F}_K} J(\Delta(q))$ also gives

$$T\tau < J\big(\Delta^{(0)}\big) - J_{\min}.$$

This is an upper bound on accepted steps, not a claim that the attained state minimizes J.

Conditional one-swap local optimality. Let $\mathcal{N}_1(q)$ contain every admissible state formed by removing one current nonzero coordinate, adding one current zero coordinate with every admissible nonzero code, and preserving cardinality K. This is the keep/delete-add neighborhood: it does not include a recoding-only move of a current nonzero coordinate. A forward-only implementation considers only unactivated reference-zero coordinates and unreleased reference-nonzero coordinates, and cannot undo a prior activation or restore a prior release. Its neighborhood is generally a strict subset of $\mathcal{N}_1(q)$. Suppose an uncapped algorithm evaluates all states in $\mathcal{N}_1(q)$ at every iteration and stops only when no $q' \in \mathcal{N}_1(q)$ satisfies $J(q') < J(q) - \tau$. Its terminal state $\widehat{q}$ then satisfies

$$J(\widehat{q}) \leq J(q') + \tau, \qquad \forall q' \in \mathcal{N}_1(\widehat{q}). \tag{26}$$

This follows immediately by negating the stopping condition.

ReAlignQ uses restricted candidate pools and an approximate, potentially capped search. Therefore, Equation equation 26 is a conditional result for an exhaustive variant, not a guarantee of the implemented solver. The main algorithm claims only candidate-space feasibility and accepted-step surrogate descent; neither property implies global optimality or monotonic improvement of the true loss.

A.4 AUXILIARY LOCAL-SURROGATE FIDELITY

The gradient-Gram surrogate is not generally equal to the true calibration loss change. This subsection separates error due to curvature mismatch from the Taylor remainder. Here J_λ makes explicit the damping parameter that is implicit in J in the preceding subsections. For ReAlignQ, $C = G = N^{-1}U^\top U$ when the diagonal statistic satisfies $F_i = G_{ii}$; when a separately collected diagonal is used, C includes the correction above.

Restrict the calibration loss to an M-dimensional candidate subspace and write

$$g = \nabla \mathcal{L}_{\mathrm{cal}}(w^0), \qquad H = \nabla^2 \mathcal{L}_{\mathrm{cal}}(w^0).$$

Let $C = C^\top$ be the curvature matrix used by the surrogate, so that

$$J_\lambda(\Delta) = g^\top \Delta + \frac{1}{2}\Delta^\top C \Delta + \lambda \|\Delta\|_2^2.$$

Define the regularized true transition objective

$$T_\lambda(\Delta) = \mathcal{L}_{\mathrm{cal}}(w^0 + \Delta) - \mathcal{L}_{\mathrm{cal}}(w^0) + \lambda \|\Delta\|_2^2.$$

Assume that the segment $\{w^0 + t\Delta : t \in [0,1]\}$ lies in a region on which the calibration loss is twice continuously differentiable and its Hessian is ρ-Lipschitz:

$$\left\|\nabla^2 \mathcal{L}_{\mathrm{cal}}(x) - \nabla^2 \mathcal{L}_{\mathrm{cal}}(y)\right\|_{\mathrm{op}} \le \rho \|x - y\|_2.$$

For transitions involving at most r coordinates, define the restricted curvature mismatch

$$\epsilon_r = \sup_{\substack{v \neq 0 \\ \|v\|_0 \le r}} \frac{\left|v^\top (H - C) v\right|}{\|v\|_2^2}. \tag{27}$$

Theorem 2 (Taylor–curvature fidelity)**.** *Under the assumptions above, every admissible transition satisfying $\|\Delta\|_0 \le r$ obeys*

$$|T_\lambda(\Delta) - J_\lambda(\Delta)| \le \frac{\epsilon_r}{2}\|\Delta\|_2^2 + \frac{\rho}{6}\|\Delta\|_2^3. \tag{28}$$

Proof. Taylor's theorem with integral remainder gives

$$\mathcal{L}_{\mathrm{cal}}(w^0 + \Delta) = \mathcal{L}_{\mathrm{cal}}(w^0) + g^\top \Delta + \frac{1}{2}\Delta^\top H \Delta + R_3(\Delta),$$

where

$$R_3(\Delta) = \int_0^1 (1 - t)\, \Delta^\top \left[\nabla^2 \mathcal{L}_{\mathrm{cal}}(w^0 + t\Delta) - H\right] \Delta \, dt.$$

Hessian Lipschitzness yields

$$|R_3(\Delta)| \le \rho \|\Delta\|_2^3 \int_0^1 t(1 - t)\, dt = \frac{\rho}{6}\|\Delta\|_2^3.$$

By Equation equation 27,

$$\left|\frac{1}{2}\Delta^\top (H - C)\Delta\right| \le \frac{\epsilon_r}{2}\|\Delta\|_2^2.$$

Finally,

$$T_\lambda(\Delta) - J_\lambda(\Delta) = \frac{1}{2}\Delta^\top (H - C)\Delta + R_3(\Delta),$$

and the triangle inequality proves Equation equation 28. □

A bounded local neighborhood. If at most k coordinates change and $|\Delta_i| \le R_{\max}$, then $\|\Delta\|_2 \le \sqrt{k} R_{\max}$, and Theorem 2 gives

$$|T_\lambda(\Delta) - J_\lambda(\Delta)| \le \frac{\epsilon_k}{2} k R_{\max}^2 + \frac{\rho}{6} k^{3/2} R_{\max}^3. \tag{29}$$

This corollary describes fidelity through the realized physical edit and does not impose an additional algorithmic restriction. Damping penalizes the edit norm in the surrogate, but without further objective conditions it does not by itself prove that the selected edit is small.

Ranking and descent certificates. Let

$$B(\Delta) = \frac{\epsilon_r}{2}\|\Delta\|_2^2 + \frac{\rho}{6}\|\Delta\|_2^3.$$

The following certificates require that every transition mentioned has support size at most r and that its segment from w^0 lies inside the ρ-Lipschitz region assumed above. For two transitions $\Delta^{(a)}$ and $\Delta^{(b)}$, the condition

$$J_\lambda(\Delta^{(a)}) + B(\Delta^{(a)}) < J_\lambda(\Delta^{(b)}) - B(\Delta^{(b)})$$

certifies that $T_\lambda(\Delta^{(a)}) < T_\lambda(\Delta^{(b)})$. Likewise, for a current edit Δ and a proposed edit Δ',

$$J_\lambda(\Delta') - J_\lambda(\Delta) < -B(\Delta') - B(\Delta)$$

certifies $T_\lambda(\Delta') < T_\lambda(\Delta)$. Mere surrogate descent does not imply either conclusion without such an approximation margin.

Approximate-surrogate comparison. Under the same path and support assumptions, suppose an optimizer returns $\widehat{\Delta}$ satisfying

$$J_\lambda(\widehat{\Delta}) \le J_\lambda(\Delta^\star) + \eta$$

for a comparison transition $\Delta^\star$ and $\eta \ge 0$. Then

$$T_\lambda(\widehat{\Delta}) \le T_\lambda(\Delta^\star) + \eta + B(\widehat{\Delta}) + B(\Delta^\star).$$

This is a conditional comparison bound. The implemented capped candidate search does not provide a known value of η, so it is not a near-optimality guarantee for ReAlignQ.

Unregularized-loss corollary. The fidelity theorem also gives

$$\left|\mathcal{L}_{\mathrm{cal}}(w^0 + \Delta) - \mathcal{L}_{\mathrm{cal}}(w^0) - J_\lambda(\Delta)\right| \le \left(\frac{\epsilon_r}{2} + \lambda\right)\|\Delta\|_2^2 + \frac{\rho}{6}\|\Delta\|_2^3.$$

Thus, the damping term is part of the optimization surrogate; it does not by itself certify a decrease in the unregularized calibration loss.

Gradient Gram is not the Hessian. The empirical gradient Gram

$$G = \frac{1}{N}U^\top U$$

is positive semidefinite and has rank at most $\min(N, M)$, but generally $G \neq H$. For example, for $\ell(w) = \frac{1}{2}w^2$ at $w^0 = 0$, the gradient vanishes and hence $G = 0$, whereas $\nabla^2\ell(0) = 1$. The matrix G should therefore be described as an empirical gradient-Gram curvature surrogate, not as the true Hessian.

Scope. The fidelity result is local to the calibration loss, candidate subspace, and bounded transition path. It does not establish full-model cross-layer fidelity, global optimality of the approximate solver, or guaranteed improvement in test perplexity or downstream accuracy.

Empirical diagnostics. The quantities ϵ_r and ρ need not be numerically small merely because the theorem holds. Useful diagnostics compare predicted and measured calibration-loss changes, estimate restricted Rayleigh errors with Hessian–vector products on small candidate subspaces, and trace approximation error along a fixed transition direction at multiple step sizes. When $C \neq H$, the discrepancy generally contains a quadratic curvature-mismatch term before the cubic Taylor remainder; an observed error curve alone does not identify the remainder coefficient.

A.5 Conditional Finite-Class Calibration Generalization

This final subsection is an auxiliary conditional result, not a fourth main-paper theorem. It concerns population generalization of the empirical gradient-Gram surrogate, rather than fidelity of that surrogate to the true loss. Throughout this subsection we specialize to

$$C = G = \frac{1}{N}U^\top U.$$

Consequently, the result does not directly cover the separately corrected curvature

$$C = G + \mathrm{Diag}\Big(\widetilde{F} - \mathrm{diag}(G)\Big)$$

used when $\widetilde{F}_i \neq G_{ii}$. Extending the argument to that case requires adding $\frac{1}{2}\sum_i(\widetilde{F}_i - G_{ii})\Delta_i^2$ consistently to the empirical and population objectives and controlling the sampling, independence, and boundedness of $\widetilde{F}$. None of the claims below implicitly supplies that control.

Fixed transition class. Let $X_1, \ldots, X_N \overset{\text{i.i.d.}}{\sim} \mathcal{P}$ be calibration sequences and let $\mathcal{F}$ be a finite set of admissible physical candidate transitions. For a sequence gradient $u(X) = \nabla_{\mathcal{C}}\ell(X; w^0)$, define

$$\varphi_\Delta(X) = u(X)^\top\Delta + \frac{1}{2}\big(u(X)^\top\Delta\big)^2,$$

$$\widehat{R}_N(\Delta) = \frac{1}{N}\sum_{t=1}^{N}\varphi_\Delta(X_t) + \lambda\|\Delta\|_2^2, \qquad R(\Delta) = \mathbb{E}_{X\sim\mathcal{P}}[\varphi_\Delta(X)] + \lambda\|\Delta\|_2^2.$$

The class $\mathcal{F}$, the parent, scales, codebook, and damping parameter must be fixed before observing $X_1, \ldots, X_N$, or constructed from data independent of these sequences.

Theorem 3 (Conditional finite-class generalization)**.** *Suppose every $\varphi_\Delta(X)$, $\Delta \in \mathcal{F}$, lies almost surely in an interval of width at most $W > 0$. Then, for every $\delta \in (0, 1)$, with probability at least $1 - \delta$,*

$$\sup_{\Delta\in\mathcal{F}}\Big|\widehat{R}_N(\Delta) - R(\Delta)\Big| \le W\sqrt{\frac{\log(2|\mathcal{F}|/\delta)}{2N}}.$$

Proof. For a fixed Δ, Hoeffding's inequality gives

$$\Pr\Big(\Big|\widehat{R}_N(\Delta) - R(\Delta)\Big| \ge \varepsilon\Big) \le 2\exp\left(-\frac{2N\varepsilon^2}{W^2}\right).$$

A union bound over the fixed finite class $\mathcal{F}$, followed by setting the right-hand side to δ, proves Equation equation 3. □

If $W = 0$, each response $\varphi_\Delta(X)$ is almost surely constant for its fixed Δ, so $\widehat{R}_N(\Delta) = R(\Delta)$ simultaneously for the finite class with probability one; this degenerate case is handled directly rather than through the displayed Hoeffding exponent.

Approximate empirical optimization. If $\widehat{\Delta}$ is an η-approximate empirical minimizer, then on the same event,

$$R(\widehat{\Delta}) - \min_{\Delta\in\mathcal{F}} R(\Delta) \le \eta + 2W\sqrt{\frac{\log(2|\mathcal{F}|/\delta)}{2N}}.$$

Indeed, applying the uniform deviation once to $\widehat{\Delta}$, using its η-approximate empirical optimality, and applying the deviation again to a population minimizer gives the result. The implemented bounded greedy search does not supply a known global η, so this inequality is not an optimizer guarantee for ReAlignQ.

Response bounds and class size. One sufficient route to the interval-width assumption is a uniform bound $|u(X)^\top\Delta| \le \Gamma$. Since $\varphi_\Delta(X) = a + a^2/2$ for $a \in [-\Gamma, \Gamma]$, an admissible common width is

$$\overline{W}(\Gamma) = \begin{cases} 2\Gamma, & 0 \le \Gamma \le 1, \\ (\Gamma+1)^2/2, & \Gamma \ge 1. \end{cases}$$

For example, $\|u(X)\|_\infty \le G_\infty$ and $\|\Delta\|_1 \le D_1$ imply $\Gamma \le G_\infty D_1$. Observed finite gradient responses do not, by themselves, prove this almost-sure assumption.

For activation and release candidate sets of sizes $h = |\mathcal{C}^{\text{zero}}|$ and $s = |\mathcal{C}^{\text{nonzero}}|$, deficit $d = K - n_0$, and m_i admissible nonzero codes at reference-zero coordinate i, the number of ideal feasible support-code states is

$$|\mathcal{F}_d| = \sum_{b=\max(0,-d)}^{\min(s,h-d)} \binom{s}{b} \sum_{\substack{\mathcal{A}\subseteq\mathcal{C}^{\text{zero}} \\ |\mathcal{A}|=d+b}} \prod_{i\in\mathcal{A}} m_i.$$

If $m_i \le m$, then

$$|\mathcal{F}_d| \le \sum_{b=\max(0,-d)}^{\min(s,h-d)} \binom{s}{b}\binom{h}{d+b} m^{d+b} \le 2^s(1+m)^h.$$

The bound makes explicit that candidate restriction reduces statistical complexity only logarithmically while potentially excluding better transitions.

For L linear layers with independently fixed target cardinalities, the model-level feasible class is the Cartesian product $\mathcal{F}_{\text{model}} = \prod_{l=1}^{L} \mathcal{F}_l$, so that

$$\log|\mathcal{F}_{\text{model}}| = \sum_{l=1}^{L} \log|\mathcal{F}_l|.$$

This is a counting identity, not a separability assumption for the full-model loss or its gradient-Gram curvature.

Same-sample selection caveat. When the same calibration data is used both to construct the shortlist and to evaluate its transitions, the realized class is data-dependent. In that case, Equation equation 3 cannot be applied merely by substituting the realized shortlist size. A valid route requires sample splitting, a union bound over a predetermined ambient transition class, or a separate data-dependent complexity argument. Define the regularized population true-loss transition objective as

$$\mathcal{R}_{\text{true},\lambda}(\Delta) = \mathbb{E}_{X\sim\mathcal{P}}\left[\ell(X; w^0+\Delta) - \ell(X; w^0)\right] + \lambda\|\Delta\|_2^2.$$

Candidate restriction decreases the logarithmic estimation term when the class is validly fixed, but can also exclude better transitions and thereby introduce candidate-restriction bias. For a candidate class $\mathcal{F}_{\text{cand}}$ contained in a larger admissible class $\mathcal{F}_{\text{full}}$, the resulting excess true objective decomposes exactly as

$$\begin{aligned}
&\mathcal{R}_{\text{true},\lambda}(\widehat{\Delta}) - \min_{\Delta\in\mathcal{F}_{\text{full}}} \mathcal{R}_{\text{true},\lambda}(\Delta) \\
&= \left[\mathcal{R}_{\text{true},\lambda}(\widehat{\Delta}) - \min_{\Delta\in\mathcal{F}_{\text{cand}}} \mathcal{R}_{\text{true},\lambda}(\Delta)\right] \\
&\quad + \left[\min_{\Delta\in\mathcal{F}_{\text{cand}}} \mathcal{R}_{\text{true},\lambda}(\Delta) - \min_{\Delta\in\mathcal{F}_{\text{full}}} \mathcal{R}_{\text{true},\lambda}(\Delta)\right].
\end{aligned}$$

The first bracket is estimation and optimization error within the retained class; the second is candidate-restriction bias. This result neither removes that bias nor implies population true-loss, test-perplexity, or zero-shot guarantees.

Composition with true-loss approximation. If an additional, uniform bound

$$\sup_{\Delta\in\mathcal{F}} |\mathcal{R}_{\text{true},\lambda}(\Delta) - R(\Delta)| \le \alpha$$

holds for the regularized population true-loss transition objective, then the approximate empirical optimizer has excess population true objective at most

$$\eta + 2W\sqrt{\frac{\log(2|\mathcal{F}|/\delta)}{2N}} + 2\alpha.$$

Such an α is an additional assumption: the local Taylor–curvature result above controls calibration-loss transitions, not this population quantity automatically.

B SPARSE-QUANTIZED OPERATOR IMPLEMENTATION

B.1 SCOPE AND INTERFACE BOUNDARY

The operator backend evaluates a sparse-quantized linear transformation

$$Y = XW^\top, \qquad W \in \mathbb{R}^{M\times K}, \quad X \in \mathbb{R}^{B\times K}.$$

It is designed to accept fine-grained nonzero supports rather than requiring an $N{:}M$ pattern or the removal of complete blocks. The implementation uses 8×8 tiles as storage and decoding units, but retained positions inside each tile may remain element-wise and irregular.

Bitmap preservation, physical nonzero-support preservation, and quantized-value preservation are distinct properties. The exact-bitmap paths described below retain every supplied mask bit without uniformization. A retained value may nevertheless quantize to integer code zero if the supplied floating-point weight and operator scale are incompatible. Exact preservation of a support-code state therefore additionally requires compatible codebooks and scale granularity, followed by an explicit check that every retained coordinate has the intended nonzero code. In particular, the W4A16/W8A16 and W8A8 paths use different scale layouts. If an input checkpoint uses another scale grouping, converting it to the operator ABI is a separate requantization step and must not be described as exact support-code preservation without this check.

The released backend contains three execution families:

- W4A16: signed INT4 weights with FP16 activations;
- W8A16: signed INT8 weights with FP16 activations;
- W8A8: signed INT8 weights and signed INT8 activations.

The two activation-precision families use separate metadata ABIs and cannot be interchanged.

B.2 BITMAP REPRESENTATION OF THE NONZERO SUPPORT

Each weight matrix is partitioned into 8×8 tiles. For a tile, bit $8r + c$ of a 64-bit bitmap records whether local position (r, c) belongs to the supplied nonzero support. Retained quantized values are stored in increasing bitmap-bit order. The index of a retained value at bit position b is therefore

$$\mathrm{rank}(b) = \mathrm{popcount}\left(\mathrm{bitmap}\,\&(2^b - 1)\right).$$

This representation avoids a coordinate stream for individual nonzeros. Offsets locate the compressed-value segment associated with a tile or a larger tile group.

An all-zero tile is represented by a zero bitmap and an empty value segment. A fully populated tile is represented by a bitmap whose 64 bits are set. Neither case requires a special support convention.

B.3 W4A16 AND W8A16 METADATA

The W4A16 and W8A16 paths use a hierarchical layout:

$$64 \times 64 \text{ global tile} \longrightarrow 16 \times 64 \text{ warp-row group} \longrightarrow 8 \times 8 \text{ bitmap tile}.$$

For each 8×8 tile, the metadata stores a bitmap, a FP16 tile scale, and a variable-length compressed value segment. Global-tile and warp-row offsets allow each cooperative thread array and warp to locate the relevant segment without scanning preceding tiles.

W4A16 packs eight signed four-bit values into one 32-bit word. W8A16 stores one signed eight-bit value per byte. In both cases, an integer weight is multiplied by its tile scale while the sparse fragment is decoded into registers. The resulting FP16 fragment and the FP16 activation fragment are then consumed by FP16 Tensor Core matrix multiplication with FP32 accumulation.

The corresponding data flow is

compressed values, bitmap, scales, and activations $\longrightarrow$ shared-memory staging $\longrightarrow$ bitmap-rank decoding and weight dequant

The implementation uses aligned asynchronous copies, shared-memory double-buffering for activation tiles, register-level fragment buffering, and a swizzled shared-memory layout. These mechanisms change data movement but do not alter the supplied support.

B.4 W8A8 Metadata and Scale Semantics

The W8A8 path has an independent row-major tile ABI. Its principal fields are

- `bitmap` $\in$ `int64`^{T}: one 64-bit word per tile;
- `compressed` $\in$ `int8`^{P}: retained signed INT8 values in tile and bitmap order;
- `tile_offsets` $\in$ `int32`$^{T+1}$: compressed-stream prefix offsets for the nonuniform path;
- `row_scales` $\in$ `float16`^{M}: one weight scale per output row;
- $X_q \in$ `int8`$^{B\times K}$: quantized activations;
- activation scales: either one tensor scale or one scale per token.

The released packer derives

$$s_{w,m} = \frac{\max\{\max_k |W_{m,k}|, 10^{-8}\}}{127}, \qquad W_{q,m,k} = \mathrm{clip}\left(\mathrm{round}(W_{m,k}/s_{w,m}), -127, 127\right).$$

Only values whose bitmap bits are set are written to the compressed stream.

For output row m and flattened token row b, the implemented computation is

$$Y_{b,m} = \left(\sum_{k=1}^{K} X_{q,bk} W_{q,mk}\right) s_{x,b} s_{w,m}.$$

The inner product is accumulated in signed 32-bit integer registers using $\mathrm{s8}\times\mathrm{s8}\to\mathrm{s32}$ Tensor Core operations. Weight and activation scales are applied in the output stage.

W8A8 uses one weight scale per output row rather than one scale per 8×8 tile. A Tensor Core instruction accumulates products from multiple K positions before dequantization; different K segments therefore cannot be assigned independent tile scales after they have entered the same integer accumulator. A row scale remains constant along K and can be applied after accumulation. Consequently, W8A8 row scales must not be replaced by the tile scales used by the W4A16/W8A16 ABI.

B.5 Nonuniform and Uniform W8A8 Paths

Exact-bitmap nonuniform-support path. The default path for an externally supplied unstructured mask preserves every bitmap bit without snapping the mask to a uniform per-tile count. This is "lossless" only with respect to mask uniformization; the encoded-reference check below separately determines the numerical effect of quantization. Tiles may contain different numbers of retained values. To retain aligned bulk transfers, eight consecutive tile columns in one tile row are grouped into an octet. Values remain variable-length within the octet, while the octet's total compressed span is padded to a 16-byte boundary. Padding bytes have no corresponding set bitmap bit and therefore do not contribute to the output.

The kernel stages the bitmap, padded compressed stream, and octet-local offsets in shared memory. Each tile recovers its value address as a local offset plus the bitmap rank. This path is exposed through the W8A8 nonuniform `forward` entry and is used by the exact-mask module constructor.

Uniform-nonzero path. The separate `forward_fast` entry assumes that every 8×8 tile contains the same number of stored values. Under this assumption, a tile's compressed-stream base is an affine function of its tile index, so `tile_offsets` are unnecessary.

This path is bitmap-preserving only when the supplied mask already satisfies the uniform-nonzero condition. Snapping an arbitrary mask to this layout by dropping or inserting positions changes the bitmap and is not equivalent to exact-bitmap execution of a ReAlignQ-selected state. The implementation therefore keeps the uniform and nonuniform paths as separate interfaces.

Correctness oracle. A third entry, `forward_naive`, accepts variable tile offsets and serves as an independently structured correctness oracle. It is not the default execution path.

B.6 W8A8 KERNEL DATA FLOW

Each cooperative thread array computes a 64×64 output tile and advances over K in blocks of 64. Compressed weights and bitmap words are staged with aligned asynchronous copies and ping-pong shared-memory buffers. For each required group of K positions, the kernel:

1. reads the corresponding bitmap bits;
2. computes the initial compressed-value rank with population count;
3. loads a compact byte window from the staged value stream;
4. reconstructs sparse INT8 fragments with byte permutation;
5. loads the matching INT8 activation fragment;
6. applies signed INT8 Tensor Core matrix multiplication; and
7. applies row and activation scales in the output stage.

The loop order decodes one sparse weight fragment before reusing it across multiple activation/output subtiles. A coordinated permutation of K slots is applied to both operands so that activation fragments can be loaded in aligned groups without changing the mathematical inner product.

When Split-K is selected, the K dimension is partitioned across multiple cooperative thread arrays. Partial FP16 outputs are written to a workspace and accumulated by a separate FP32 reduction. Split-K is therefore part of the execution schedule, not part of the sparse representation.

B.7 PER-TOKEN ACTIVATION QUANTIZATION

For per-token W8A8 activation quantization, flattened token row b uses

$$s_{x,b} = \frac{\max\{\max_k |X_{b,k}|, 10^{-8}\}}{127}, \qquad X_{q,b,k} = \mathrm{clip}\left(\mathrm{round}(X_{b,k}/s_{x,b}), -127, 127\right).$$

A dedicated CUDA kernel assigns one activation row to a cooperative thread array, reduces its absolute maximum, and emits both the INT8 row and its FP16 token scale. The quantized activation is produced once and may be reused by all output-row tiles. The nonuniform W8A8 matrix-multiplication kernel accepts the token-scale vector and applies $s_{x,b}s_{w,m}$ directly in its output stage, avoiding a separate output-rescaling operation.

The uniform `forward_fast` interface uses a scalar activation scale in the released ABI. Per-token activation scales are supported by the nonuniform W8A8 path. A per-tensor activation scale remains available as a fallback.

B.8 RUNTIME CONTRACTS AND CORRECTNESS CHECKS

The Python and C++ bindings verify tensor dtypes, contiguity, dimensions, and device placement. All operands and metadata tensors must reside on the same CUDA device, and a device guard selects that device before launching the kernel.

The released W8A8 linear module does not fuse a bias term. Its constructor accepts only bias-free linear layers; when a model contains a bias, that bias must remain outside the sparse matrix-multiplication kernel and be applied separately.

The current W8A8 cooperative-thread-array geometry requires M, K, and the flattened token dimension B to be multiples of 64. Unsupported matrix dimensions can be zero-padded before execution and cropped afterward, provided that padded positions are excluded from the reported support semantics.

Split-K values are checked before launch. The number of K tiles assigned to a group and the padded final group must satisfy the kernel's partition rule; violating this rule is rejected by the wrapper rather than treated as a valid schedule.

Correctness is checked against an encoded reference rather than against the original dense FP16 matrix alone. The reference reconstructs the quantized sparse weight from the bitmap, compressed values, offsets, and scales, then computes the corresponding dense operation. Additional checks

compare the nonuniform W8A8 kernel with the independent oracle for every supported Split-K configuration and exercise memory safety with CUDA sanitization. These tests separate kernel correctness from pruning and quantization error.

B.9 Implementation Limitations

The released implementation has been validated for NVIDIA RTX 6000D (`sm_120`); the same behavior is not claimed for untested GPU architectures. Its tile and cooperative-thread-array constraints may require padding for other model dimensions.

The uniform W8A8 entry cannot consume an arbitrary nonuniform mask without a bitmap-changing conversion. The nonuniform entry should be used whenever the input bitmap must be retained exactly; physical nonzero-support equivalence still requires the code-and-scale check described above.

The released W4A16/W8A16 kernel can produce occasional non-finite outputs for very long flattened token dimensions. Such inputs require finite-output checking and smaller host-side chunks in this version. This limitation is specific to the released implementation and is not hidden by the binding.

W4A4 and FP8 are not supported execution paths. System-level evaluation of a packed ReAlignQ checkpoint is outside the scope of this implementation appendix.

C Supplementary Experimental Details and Results

C.1 Evaluation Protocol and Reproducibility

All perplexity evaluations use WikiText-2 raw-v1 with a fixed 65,536-token prefix, yielding 65,504 predicted tokens in 32 non-overlapping windows of length 2,048. Zero-shot evaluation uses PIQA, BoolQ, HellaSwag, ARC-Easy, ARC-Challenge, and WinoGrande with zero in-context examples. Every ReAlignQ result retains the quantization scales and target support size of its corresponding compressed model.

C.2 Algorithm and Solver Configuration

The following pseudocode summarizes the implementation used for the reported results. Both gradient passes evaluate the full model at the common reference state W^0. Candidate screening, signature interactions, and support exchanges are then performed separately for each linear weight matrix. The algorithm does not form terms that model interactions across matrices or recompute gradients after each matrix edit.

For an activation candidate, the code routine evaluates the floor and ceiling of its continuous target, the smallest representable positive and negative codes, and the positive and negative code bounds. It selects the code with the smallest marginal surrogate value. A coordinate that is nonzero in the reference state is not recoded. It either retains its original code or is released. These settings define the solver used in the main results and the matched ablations unless a table states otherwise.

For W4A16 and W8A8, the run manifests retain the same calibration protocol and search procedure. The codebook and scale layout follow the corresponding compressed-model format.

C.3 Offline Refinement Cost

Table 8 reports the added offline cost of ReAlignQ for the Llama3-8B EIWR W8A16 parent at 70% sparsity. The run uses 256 calibration sequences. The table reports GPU compute time and peak GPU memory. It excludes PPL and zero-shot evaluation.

The aggregate statistics pass dominates GPU compute time. Candidate signatures are retained only after screening. They are used during calibration and are not needed for inference. ReAlignQ preserves the sparse layout and quantization format of the parent. Deployment storage is reported separately in Table 11.

Algorithm S1: ReAlignQ

Require: reference codes Z^0, fixed scales S, support targets $\{K_l\}_{l=1}^{L}$, calibration data $\mathcal{D}_{\mathrm{cal}}$, candidate budgets, damping λ, exchange cap E, and acceptance margin τ.
Ensure: final codes q with $\|q_l\|_0 = K_l$ and $q_{l,i} \in \mathcal{Q}_{l,i}^{\mathrm{adm}}$ for every matrix l.

1. Evaluate full model gradients on $\mathcal{D}_{\mathrm{cal}}$ at $W^0 = S \odot Z^0$. Aggregate g_i and F_i for every linear weight coordinate.
2. For each linear weight matrix l, screen coordinates that are zero in the reference state for activation and coordinates that are nonzero for release. Form $\mathcal{C}_l^{\mathrm{zero}}$, $\mathcal{C}_l^{\mathrm{nonzero}}$, and $\mathcal{C}_l = \mathcal{C}_l^{\mathrm{zero}} \cup \mathcal{C}_l^{\mathrm{nonzero}}$.
3. Repeat the full model gradient pass at W^0. Retain U_l only for candidates in $\mathcal{C}_l$.
4. For each linear weight matrix l:
   (a) Construct a feasible state with $\|q_l\|_0 = K_l$ using coupled marginals. Initialize its physical edit Δ_l and response $r_l = U_l \Delta_l$.
   (b) For at most E iterations, form bounded frontiers of activation and release candidates. For each pair in these frontiers, choose the admissible incoming code that minimizes $\Delta J_{i,j}^{\mathrm{pair}}(q \mid \Delta_l)$.
   (c) If the best pair satisfies $\Delta J_{i,j}^{\mathrm{pair}} < -\tau$, apply its activation and release, then update Δ_l and r_l. Otherwise, stop processing matrix l.
5. Materialize all matrix edits using the original scales. Verify support cardinality and the legality of all incoming codes.

Table 6: Pseudocode for ReAlignQ. Candidate search and interaction evaluation occur within each linear weight matrix. Existing nonzero coordinates retain their reference code or are released. Code selection applies to coordinates activated from zero.

Table 7: Solver configuration for the primary W8A16 experiments. The reported damping ratio is specified in each run manifest before checkpoint evaluation and is fixed across the primary cells.

Component	Setting
Calibration data	256 C4 sequences of length 2,048, split seed 0
Quantization	Fixed parent scales and signed INT8 codebook
Incoming code range	$\lvert q\rvert \le 2$ for newly activated coordinates
Candidate signatures	Coupled gradient signatures for each sample
Feasible-state construction	Three fixed point updates
Pair frontier	At most 512 activation candidates and 128 release candidates
Exchange cap	Eight accepted activation and release exchanges per matrix
Acceptance tolerance	Pair score $< -10^{-7}$
Applied damping	$\lambda_l = r\,\mathrm{median}_{i \in \mathcal{C}_l} \lvert g_i s_i\rvert$ for matrix l
Damping ratios built	$10^6, 2{\times}10^6, 3{\times}10^6, 5{\times}10^6, 10^7$
Reported damping ratio	$3{\times}10^6$

C.4 Additional Compression Regimes

ReAlignQ reduces PPL at all three W4A16 sparsity levels. The zero-shot average is unchanged at 50% sparsity and improves at 60% and 70%, indicating that the recovery persists in this more aggressive quantization regime.

C.5 Surrogate Pair-Score Fidelity

We evaluate whether the interaction surrogate ranks legal support exchanges by their realized held-out loss improvement. Starting from the SparseGPT+GPTQ Llama3-8B W8A16 model at 70% sparsity, we pre-sample 128 legal activation–release pairs. Each pair is independently applied to the same compressed model and evaluated on disjoint C4 tokens. The predicted improvement is defined as the negative surrogate change, $-\Delta J_{i,j}^{\mathrm{pair}}$, so that larger values indicate more favorable exchanges. The realized improvement is the held-out loss reduction after applying the pair.

Table 8: Offline cost of ReAlignQ on Llama3-8B with the EIWR W8A16 70% parent. GPU compute time includes the gradient and coupled-search stages. Signature storage is calculated for the retained $256 \times 1{,}084{,}240$ float32 signature matrix. It does not include temporary shard files.

Quantity	Value
GPU compute time	3.97 GPU-hours
Peak GPU memory	37,152 MiB
Screened candidates	1,084,240
Retained signature storage	1.03 GiB

Table 9: Additional paired W4A16 recovery results on Llama3-8B. W4 rows use the compressed model's realized post-quantization support size and pass the calibration gate. P/R denotes compressed model / ReAlignQ.

Parent	Sparsity	PPL P/R↓	ZS P/R↑	Add/Del	Gate
SparseGPT+GPTQ	50%	12.33 / 11.69	65.3 / 65.3	7 / 7	pass
SparseGPT+GPTQ	60%	24.73 / 22.81	58.4 / 60.4	19 / 19	pass
SparseGPT+GPTQ	70%	86.86 / 71.69	44.8 / 48.7	19 / 19	pass

The positive rank correlation and quartile separation indicate that higher predicted-improvement scores are associated with larger independently measured held-out loss improvements.

C.6 Exact-Mask Deployment

Both compressed W8A8 systems use 3.27 GiB of static storage, compared with 14.96 GiB for Dense FP16. Exact-mask verification reports zero dropped nonzeros and packed PPL agrees with the HuggingFace evaluation.

C.7 Additional Analyses

Calibration-size sensitivity. For the Llama3-8B EIWR W8A16 70% baseline, we hold the reference scales, codebook, target support, solver, and evaluator fixed while varying only the number of calibration samples. Every completed point uses the same 65,536-token WikiText-2 prefix and exact-support verification. Table 12 reports the four-point sweep; the small variation is non-monotonic, with the 64-sample point worse than both adjacent points. The largest difference from the 256-sample reference is 0.3341 PPL. No calibration-size point was selected using downstream test performance.

Disjoint calibration holdout. Table 13 evaluates the Llama3-8B EIWR W8A16 70% baseline and Full ReAlignQ on a disjoint C4 holdout. The improvement agrees with WikiText-2, providing no indication of obvious calibration overfitting.

Table 10: Pair-score ranking fidelity for 128 legal SparseGPT+GPTQ exchanges on Llama3-8B under W8A16 quantization and 70% sparsity. Higher values are better. Values in brackets are 95% bootstrap confidence intervals.

Metric	Result
Spearman rank correlation	0.36 [0.18, 0.51]
Improvement-sign accuracy	66%
Top-minus-bottom true improvement	+0.0041 [0.0012, 0.0068]

Table 11: Exact-mask W8A8 deployment on Llama3-8B with a SparseGPT+GPTQ 70% baseline. Packed PPL is evaluated under the same WikiText-2 protocol as the HuggingFace checkpoint. Static storage excludes dynamic activations and allocator cache.

System	HF PPL↓	Packed PPL↓	Prefill tok/s↑	Static Storage (GiB)↓
Dense FP16	6.32	N/A	8406.0	14.96
SparseGPT+GPTQ baseline	55.27	55.35	6642.4	3.27
SparseGPT+GPTQ + ReAlignQ	34.30	34.12	6638.6	3.27

Table 12: Fixed-protocol calibration-size sensitivity for Llama3-8B EIWR W8A16 at 70% sparsity. Lower WikiText-2 PPL is better.

Samples	WikiText-2 PPL↓	Δ vs. 256
32	25.6367	+0.0684
64	25.9025	+0.3341
128	25.5836	+0.0153
256	25.5683	+0.0000

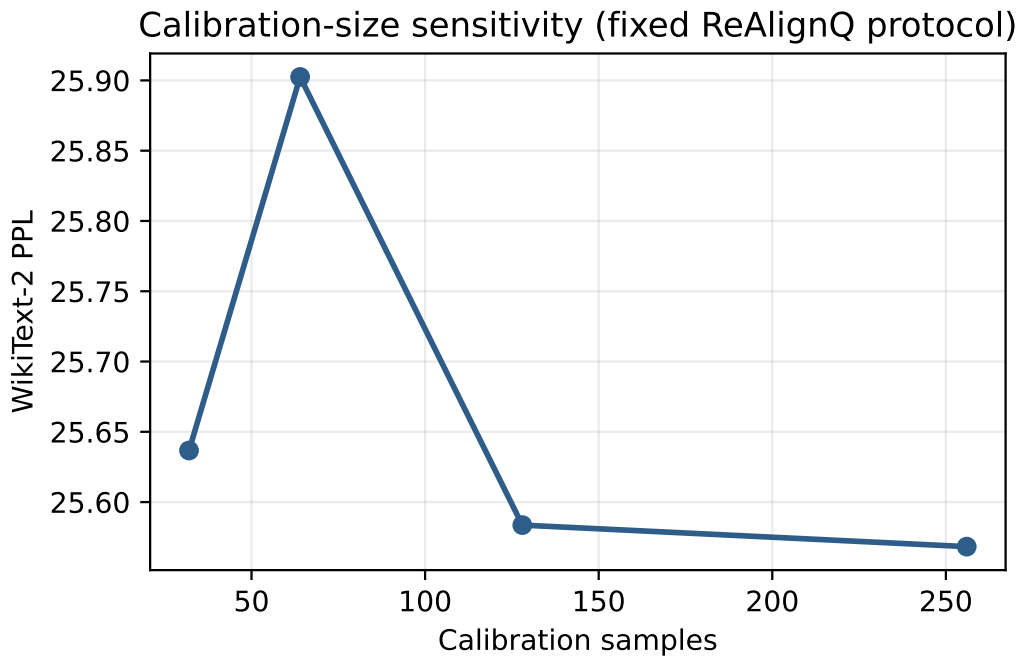

Figure 2: Calibration-size sensitivity under the fixed protocol in Table 12. Only the number of calibration samples varies.

Table 13: Disjoint C4 holdout for Llama3-8B EIWR W8A16 at 70% sparsity. Lower PPL is better.

System	WikiText-2 PPL↓	C4 holdout PPL↓
EIWR baseline	35.480	30.576
Full ReAlignQ	25.568	23.260