QUANTFORGE: DISCOVERING RESIDUAL DECOMPOSITIONS FOR MXFP4 POST-TRAINING QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Four-bit post-training quantization can reduce the memory demands of large language models, but preserving accuracy under strict MXFP4 W4A4 requires coordinating several design choices. Coordinate transforms change block-encoding errors, which in turn affect the residuals propagated through the network. The useful algorithmic decomposition is therefore not fully known before search. LLM-driven program evolution offers a way to explore these choices, but performance scores alone do not explain which design should change next. We introduce QuantForge, a PTQ discovery system that records competing explanations, selects controls that distinguish them, and checks that successor code implements the resulting conclusions. This residual compilation guides program revisions while retaining useful programs even when their original explanations are rejected. Remeasuring the revised program reveals the next error to address. This process discovers HiRes, a fixed MXFP4 quantizer that shapes coordinates, refines legal code assignments, and recovers errors along attention and MLP paths. Each stage acts on residuals measured after the preceding stage has executed. Across seven tasks, HiRes achieves the lowest seven-model Robust Fit (0.09300) and the lowest quantized Fit-7 at 32B. In matched-budget comparisons of LLM-driven program evolution, each with 240 evaluator calls, QuantForge reaches a held-out transfer target in six of eight runs, compared with three each for textual memory and reflection memory, and one for score-only evolution, despite evaluating fewer new programs. These results show that QuantForge improves the discovery of transferable PTQ algorithms by turning controlled evidence into subsequent program changes.

1 INTRODUCTION

Large language models are increasingly used for text generation, question answering, and complex reasoning (Grattafiori et al., 2024; Yang et al., 2025). Deploying them at scale requires reducing the costs of storing and running these models. Post-training quantization (PTQ) reduces these costs without retraining (Frantar et al., 2023). Four-bit weights and activations (W4A4) reduce nominal precision fourfold relative to 16-bit formats. MXFP4 provides a standardized representation: 32 values share a power-of-two scale, allowing short blocks to adapt to local magnitudes with little scaling metadata (Rouhani et al., 2023a;b). This compact representation also couples rounding precision within each block: a scale set by large values can leave smaller values poorly resolved.

A natural response is to transform coordinates so that large values are spread across a block (Ashkboos et al., 2024; Chen et al., 2026). In MXFP4, the transformed values must still fit a sparse four-bit grid with a power-of-two shared scale (Egiazarian et al., 2026; Shao et al., 2026). The coordinate choice therefore affects both scale selection and code assignment. It also changes the input correlations used by second-order reconstruction to compensate for weight-rounding error (Frantar et al., 2023). Once these quantized operators are installed, their errors propagate through attention and interact through the products in gated MLPs (Shazeer, 2020). Accurate PTQ thus requires coordinating coordinate design, discrete encoding, and path-level correction around the realized quantized computation. These dependencies make manual algorithm design difficult: improving one step can change what the next step needs to correct.

LLM-driven program evolution can explore these dependencies by rewriting quantization operations as executable code. An evaluator tests alternative constructions under a common protocol, and

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

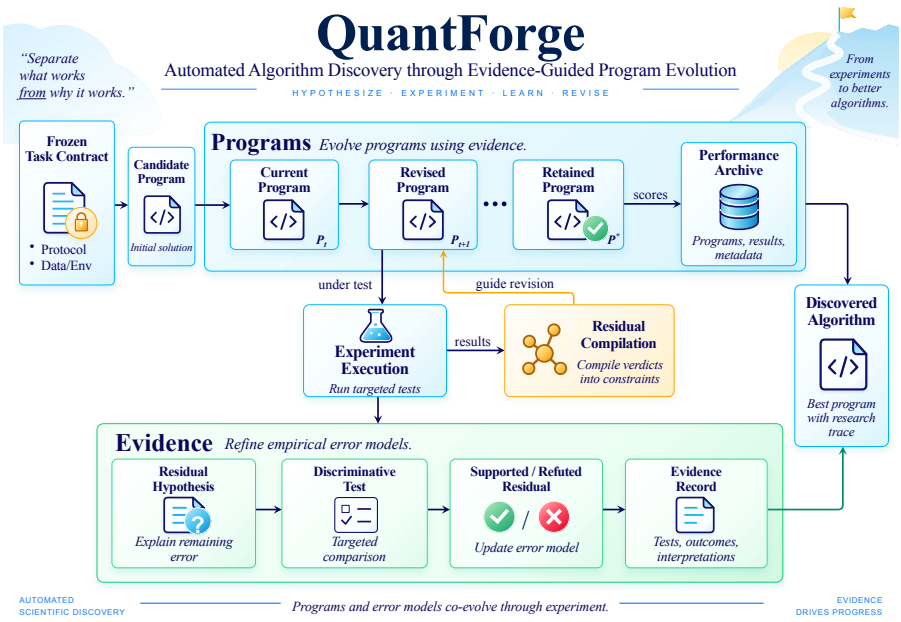

Figure 1: **Overview of QuantForge.** Program performance determines which candidates are retained; controlled experiments determine how unresolved questions guide subsequent code revisions.

successful programs inform subsequent proposals. This approach has discovered mathematical constructions and competitive algorithms (Romera-Paredes et al., 2024; Novikov et al., 2025; Liu et al., 2024). Performance is an effective selection signal, but an incomplete design signal. For example, a transform may help by spreading outliers or by improving shared-exponent selection. These explanations suggest different successors: a stronger coordinate transform in one case, revised block encoding in the other. Even when the editable interface can expand (Li et al., 2026b), the score alone cannot distinguish these directions.

We study PTQ algorithm discovery when improving one component changes the errors that subsequent components must address. QuantForge (Figure 1) uses performance to retain useful programs and controlled experiments to direct their revisions. Its defining step is to turn an experimental conclusion into a required code change and verify its implementation with an execution probe. It tracks programs in an *executable frontier* and unresolved design questions in a *residual frontier*. Proposals predict how competing explanations respond to controls; QuantForge executes the least costly distinguishing comparison and compiles its outcome into the successor's requirements. Remeasuring the revised program reveals what to address next, progressively determining the quantization algorithm's decomposition. We call this process *progressive residual factorization*.

This process yields HiRes, a fixed MXFP4 quantizer that adapts each stage to the quantized model produced by the preceding stages. Geometric shaping refines coordinates using errors from actual encoding; discrete realization selects activation codes against the installed quantized weights; structural recovery fits the remaining path errors, remeasuring the MLP residual after attention correction. These conditional updates form a deterministic quantization procedure with strong cross-model numerical quality, as evaluated below.

Our contributions are:

- **A formulation for compositional discovery.** We formulate progressive residual factorization, using controlled experiments to determine which part of a PTQ algorithm should change next. Remeasuring each revised program reveals dependencies between interventions, allowing the algorithm's decomposition to emerge during discovery.
- **An effective PTQ discovery system.** QuantForge turns experimental outcomes into required, checked code revisions while retaining useful programs independently of their proposed

explanations. Under matched 240-call budgets, it reaches the held-out target in six of eight runs, versus three each for textual and reflection memory and one for score-only evolution, despite evaluating fewer new programs.

- **A strong strict-MXFP4 quantizer.** The discovered HiRes links geometric shaping, discrete realization, and structural recovery through remeasured residuals, so later decisions account for the quantized operators produced by earlier steps. Across seven models and seven tasks, it achieves the lowest Robust Fit (0.09300), balancing average and worst-model damage, and the lowest quantized Fit-7 on Qwen3-32B.

2 Related Work

Post-training quantization. PTQ methods improve low-bit representations through reconstruction, rescaling, and coordinate transforms. GPTQ uses second-order information to compensate for weight-rounding error (Frantar et al., 2023). AWQ and SmoothQuant redistribute channel magnitudes (Lin et al., 2024b; Xiao et al., 2023), while QuaRot, SpinQuant, and DuQuant alter quantization coordinates (Ashkboos et al., 2024; Liu et al., 2025; Lin et al., 2024a). Microscaling methods address the interaction between block structure and quantization, including MR-GPTQ, BRQ, TORQ, and BATQuant (Egiazarian et al., 2026; Shao et al., 2026; Xu et al., 2026; Li et al., 2026a). WUSH constructs adaptive transforms from weight and activation statistics (Chen et al., 2026). FOCUS learns relaxed, sub-block quantization scales while retaining format-compliant dequantization scales (Yan et al., 2026). HiRes combines an error-supported geometric construction, legal discrete refinement, and recovery fitted to the realized network. We compare their numerical quality under one MXFP4 contract.

Automated algorithm and scientific discovery. FunSearch and AlphaEvolve evolve executable programs against measured performance (Romera-Paredes et al., 2024; Novikov et al., 2025). LLM-based heuristic search extends this approach through evolutionary and tree-based controllers (Liu et al., 2024; van Stein & Bäck, 2025; Zheng et al., 2025). Reflection, persistent memory, and review can guide exploration beyond scores alone (Ye et al., 2024; Agrawal et al., 2026; Liu et al., 2026; Mroueh et al., 2026); OPTScientist also supports typed optimizer programs and an extensible interface (Li et al., 2026b). Scientific agents address a broader research cycle, from generating ideas to executing experiments (Lu et al., 2024; Yamada et al., 2025; Gottweis et al., 2025). AutoDiscovery selects hypotheses to explore, and Popper turns hypotheses into executable falsification tests (Agarwal et al., 2025; Huang et al., 2025). Our focus is the step from experimental resolution to a subsequent quantizer. QuantForge records competing predictions before evaluation, selects controls that distinguish them, and checks that the conclusion changes later executable code. The resulting evidence guides both program selection and which part of the PTQ problem becomes actionable next.

3 Preliminaries: MXFP4 Post-Training Quantization

MXFP4 W4A4. Under our MXFP4 W4A4 contract, target linear operands use contiguous input-feature blocks of 32 E2M1 values with one E8M0 power-of-two scale (Rouhani et al., 2023a;b). Strict W4A4 denotes legal MXFP4 operands and stored payloads; low-dimensional recovery runs outside GEMM without adding higher-precision weight matrices. Weights are quantized once and activations dynamically. Weight-reconstruction statistics use pre-A4 calibration activations.

Second-order reconstruction. For weights $W \in \mathbb{R}^{m\times d}$ and calibration inputs $X \in \mathbb{R}^{n\times d}$, GPTQ (Frantar et al., 2023) minimizes

$$\mathcal{L}(\widehat{W}) = \frac{1}{n}\|X(W-\widehat{W})^\top\|_F^2, \quad \widehat{W} \in \mathcal{Q}, \qquad H = \frac{1}{n}X^\top X, \tag{1}$$

where $\mathcal{Q}$ is the quantized feasible set and H determines the row-wise curvature. Rounding error is compensated in the remaining input-feature columns using a stabilized factorization of H^{-1}. HiRes uses this reconstruction step in transformed coordinates.

Frozen contract. Candidates and controls share the quantization format, target modules, calibration data, evaluator, and resource limits. Experiments use simulated quantization in PyTorch (Paszke et al., 2019) to measure numerical quality. The complete execution contract appears in Appendix B.

4 QuantForge: PROGRESSIVE RESIDUAL FACTORIZATION

QuantForge maintains an *executable frontier* $\mathcal{P}_t$ of useful programs and a *residual frontier* $\mathcal{R}_t$ of unresolved design questions. Together they form the search state $S_t = (\mathcal{P}_t, \mathcal{R}_t)$ at round t (Figure 1). Controlled experiments resolve these questions, and their conclusions become requirements on later code. Remeasuring the revised program exposes the next question.

4.1 FROM PROGRAM GAIN TO ALGORITHMIC RESIDUAL

Consider an illustrative PTQ setting that permits a fixed mixed-precision budget, unlike our strict MXFP4 benchmark. Suppose keeping the largest 1% of weights at higher precision improves the score. The gain could reflect the importance of those weights or simply the added capacity. Assigning the same budget to a random 1% provides a *counterfactual control*. Improvement only for the targeted allocation supports weight identity; improvement for both supports capacity as an alternative explanation. These outcomes suggest different successors: refine outlier handling or reconsider how capacity is allocated.

The unresolved question is whether the gain comes from weight identity or added capacity. We call such a design question attached to an executable program an *algorithmic residual*: competing explanations fit the observation but imply different program changes. Performance determines whether the program remains on $\mathcal{P}_t$; controlled evidence determines whether its residual remains on $\mathcal{R}_t$ or closes. A useful program can therefore survive while its original explanation is rejected.

4.2 RESOLVING RESIDUALS WITH COUNTERFACTUALS

Before evaluation, a proposal registers explanations $\mathcal{E}$, their predicted outcomes $P_e(c)$ under condition c, and allowed controls $\mathcal{U}$. QuantForge selects the least costly subset that distinguishes every pair of explanations:

$$\mathcal{U}^* = \arg\min_{V \subseteq \mathcal{U}} \sum_{c \in V} \mathrm{cost}(c) \quad \text{s.t.} \quad \forall e \neq e',\ \exists c \in \{\text{primary}\} \cup V : P_e(c) \neq P_{e'}(c). \tag{2}$$

Here $e, e' \in \mathcal{E}$, and $\mathrm{cost}(c)$ counts the evaluation cost of condition c. The constraint requires a predicted disagreement on at least one executed condition. In the example, one random-allocation control is enough. Predictions are fixed before execution; all conditions share the evaluation contract and differ only in the specified intervention.

Control selection preserves the information needed for the next design decision. Two controls with identical predicted outcomes across the explanations need not both be executed. Conversely, an additional control is useful when it separates accounts that the first leaves indistinguishable. If the allowed controls cannot resolve that ambiguity, the proposal must formulate a better discriminator before consuming an evaluation.

4.3 COMPILING RESOLUTION INTO PROGRAM STRUCTURE

Residual compilation turns the experimental verdict into a program response. If both allocations improve, the targeted program can remain useful, but another outlier-specific rule does not resolve the capacity explanation. A successor must remove that dependence or test a sharper distinction. For selected program P_t, residual $\rho_t \in \mathcal{R}_t$, and controls $\mathcal{U}_t^*$, the transition is

$$v_t = \mathsf{Resolve}(\rho_t, \mathcal{U}_t^*), \qquad P_{t+1} = \mathsf{Compile}(P_t, v_t), \tag{3}$$

where v_t is the experimental verdict. Supported explanations permit an intervention to be extended; contradicted accounts require a revised construction. Inconclusive controls leave the residual open. The system verifies the response through changed code and a targeted execution probe. Appendix A specifies the verdicts and checks.

This separation preserves a candidate's measured gain while revising its interpretation. The retained program supplies a strong starting point, and the control supplies a more precise question for its successor. An experiment can therefore change the direction of search even when it does not change which program has the best score. Compilation preserves this information as a requirement on the next relevant computation, rather than leaving each proposal to reinterpret the same result.

4.4 Progressive Residual Factorization

The measurements motivating a program change may no longer describe the revised program. QuantForge therefore remeasures the revised program before choosing the next intervention. The new residual may call for an existing operation or a newly introduced component. Repeating this process produces *progressive residual factorization*: executed comparisons reveal which interventions are useful and which earlier program states make them relevant. The search backbone remains replaceable; residual compilation supplies the link from evidence to code. In MXFP4, the resulting hierarchy leads from geometric shaping to discrete realization and network-structured recovery, developed next.

5 HiRes: Hierarchical Residual Resolution for MXFP4

Across programs retained by QuantForge, function-preserving coordinate changes before Block32 encoding recurred as an effective intervention, building on transform-based PTQ (Ashkboos et al., 2024; Chen et al., 2026). HiRes builds on these coordinate changes by addressing the residuals measured after quantization.

5.1 Stage I: Geometric Shaping

One large coordinate can set the shared exponent for 32 values, so equivalent BF16 representations can incur different W4A4 errors. Reference coordinates P_{ref} balance activation statistics against inverse weight geometry. An interaction correction G uses errors from an actual W4A4 encoding pass to refine those coordinates. Diagonal rescaling D then balances weight and activation quantization risk in the resulting coordinate system.

Reference coordinates. We restrict the d input features in Section 3 to one contiguous block of width $q = 32$. Below, $X \in \mathbb{R}^{n\times q}$ and $W \in \mathbb{R}^{m\times q}$ denote the corresponding activation and weight matrices. We use ridge coefficients λ_A, λ_B for regularization and $\mathrm{DN}(M) = M/\det(M)^{1/q}$ to remove overall scale. The operator Cap_8 normalizes the determinant to one and limits the condition number to eight. The reference chart is

$$\begin{aligned} A &= X^\top X/n + \lambda_A I, & B &= W^\top W/m + \lambda_B I, \\ M_0 &= \tfrac{1}{2}[\mathrm{DN}(A) + \mathrm{DN}(B^{-1})], & P_{\mathrm{ref}} &= \mathrm{Cap}_8(M_0^{-1/2}). \end{aligned} \tag{4}$$

Exact regularizers and the spectral cap are specified in Appendix B.2.

The two operands respond oppositely to a coordinate change: expanding an activation direction contracts the corresponding weight direction. After determinant normalization removes overall scale, the reference chart combines these opposing geometries. Second moments describe signal energy, not the errors produced by legal rounding.

Encoding-error correction. Write $Y = XP_{\mathrm{ref}}^\top$ and $V = WP_{\mathrm{ref}}^{-1}$, and let $R_H = H_{32}/\sqrt{32}$ denote the normalized Hadamard. With strict anchor quantizers Q_A, Q_W, the activation and weight errors mapped back to the pre-Hadamard coordinates are $E_X = [Q_A(YR_H^\top) - YR_H^\top]R_H$ and $E_W = [Q_W(VR_H^\top) - VR_H^\top]R_H$. Their error–signal moments

$$K_X = E_X^\top Y/n, \qquad K_W = V^\top E_W/m \tag{5}$$

identify coordinate-pair interactions after encoding. The update weights each operand's error by the energy of the other operand, suppresses signals that disagree across sample folds or weight/activation contributions, and bounds the resulting zero-diagonal correction K. Appendix B.2 gives the pairwise construction and its fixed solver. Writing $G_0 = I + K$, the resulting map is

$$\begin{aligned} G &= |\det G_0|^{-1/q} G_0, & T &= R_H G P_{\mathrm{ref}}, \\ \widetilde{X} &= XT^\top, & \widetilde{W} &= WT^{-1}. \end{aligned} \tag{6}$$

The bound $\|K\|_2 \le 1/8 < 1$ makes $I + K$, and hence G, invertible. The paired transformation preserves $\widetilde{X}\widetilde{W}^\top = XW^\top$.

Weight–activation balance. Quantization in these coordinates reveals the remaining imbalance between activation and weight error. For each of the eight four-coordinate subgroups in a Block32 group, the two scalar risks are

$$r_A = \mathrm{tr}(G_W C_E), \qquad r_W = \mathrm{tr}(G_E C_X), \tag{7}$$

where C_X, C_E are uncentered activation and activation-error second moments, and G_W, G_E the corresponding weight and weight-error Grams, all restricted to that subgroup. Stacking, centering, and projecting the eight regularized log risk ratios gives a bounded, zero-sum proposal $u_b \in \mathbb{R}^8$ for physical block b. Strict reconstruction loss selects its accepted magnitude. Broadcasting each accepted component to its four coordinates yields $u \in \mathbb{R}^{32}$; $D = \mathrm{Diag}(2^u)$ is folded reciprocally into the operands (Appendix B.2). The risks provide a direction for redistributing quantization burden; actual reconstruction measures the outcome when both operands are encoded together. A favorable risk ratio alone is therefore insufficient to accept a step. The accepted geometry fixes the representation, while the next stage decides how that representation is realized by legal codes.

5.2 Stage II: Discrete Realization

Fixing the geometry leaves shared exponents and element codes to determine operator quality. HiRes reruns calibration after installing DT and computes GPTQ curvature from the transformed, pre-A4 activations (Frantar et al., 2023). Second-order compensation produces legal Block32 W4 weights with E8M0 exponents and E2M1 codes. Transformed input correlations govern GPTQ's compensation for weight-rounding error; reconstruction also retains the final operator's physical block boundaries.

The installed weights also guide activation rounding. For an activation block x, let q_0 be its nearest legal vector at the fixed exponent and $d_0 = q_0 - x$. With transformed BF16 weights W and decoded W4 weights $\widehat{W}$, define $G_Q = \widehat{W}^\top \widehat{W}$ and $K_Q = \widehat{W}^\top (W - \widehat{W})$. Changing coordinate i to legal value c changes the local quadratic model by

$$\Delta(i, c) = 2\delta_{ic}[G_Q d_0 - K_Q x]_i + \delta_{ic}^2 [G_Q]_{ii}, \qquad \delta_{ic} = c - [q_0]_i. \tag{8}$$

The linear term measures how a proposed code change interacts with the current output residual; the quadratic term accounts for its own error cost. The $K_Q x$ term includes the discrepancy of the installed weights, so the preferred activation code depends on the realized W4 operator. The best negative change is accepted; otherwise q_0 remains unchanged. This dynamic A4 rule keeps the exponent fixed and changes at most one legal code. Once these operators are installed, calibration can measure how their errors interact along the quantized network's computation paths.

5.3 Stage III: Structural Recovery

After installing the W4/A4 operators, HiRes fits low-dimensional corrections to the remaining attention and MLP errors. It first corrects value outputs, then remeasures the MLP residual so that the second fit reflects the corrected attention path.

For one layer and one key–value head in grouped-query attention (Ainslie et al., 2023), let $\mathbf{v}_t$ be the realized value output at calibration token t and $\mathbf{v}_t^\star$ its BF16 counterpart on the same calibration sequence. We suppress the layer and head indices below. A scalar scale a and scalar offset b are shared across tokens and head coordinates. Writing $\mathbf{b} = b\mathbf{1}$, where $\mathbf{1}$ is the head-dimensional all-ones vector, the update is

$$(a, b) = \arg\min_{a,b} \sum_t \|a\mathbf{v}_t + \mathbf{b} - \mathbf{v}_t^\star\|_2^2, \qquad \bar{\mathbf{v}}_t = a\mathbf{v}_t + \mathbf{b}. \tag{9}$$

The corrected values continue through the existing attention and quantized output projection. Each head is fitted separately against the paired BF16 trajectory.

The method then collects fresh gate and up outputs g, u from the attention-corrected model. Their gated product $m = \mathrm{SiLU}(g) \odot u$ couples errors in both MLP branches (Shazeer, 2020), where $\odot$ denotes elementwise multiplication. Three layerwise scalars α, β, η adjust the gate shift, gate scale, and up scale. To express the shift relative to gate magnitude, let σ_g be the regularized standard deviation of g, pooled over all calibration tokens and hidden coordinates in the layer. The update is

$$\bar{m} = \mathrm{SiLU}((1 + \beta)g + \alpha\sigma_g) \odot (1 + \eta)u. \tag{10}$$

Algorithm 1 HiRes quantization

1: **input:** fresh BF16 model; exact 128×2048 calibration
2: construct and fold T, D for each target linear layer
3: rerun calibration in the realized DT coordinates
4: realize legal W4 with GPTQ; install dynamic A4 (Eq. 8)
5: fit and install attention recovery (Eq. 9)
6: remeasure MLP residuals; fit and install Eq. 10
7: **return** the realized MXFP4 W4A4 model

The shift changes the operating point of the SiLU gate; the two scales adjust the factors of the product. A weighted ridge fit (Hoerl & Kennard, 1970) jointly estimates the three coefficients from a local linearization against $m^\star = \mathrm{SiLU}(g^\star) \odot u^\star$. Here $g^\star, u^\star$ are BF16 projections on the same realized layer input. The definition of σ_g and the aggregation and bounds of calibration-cell fits are specified in Appendix B.4. The corrected intermediate enters the strict W4A4 down projection.

Algorithm 1 follows three measurement dependencies. Transformed inputs determine reconstruction curvature; installed W4 weights determine the activation-refinement objective; and attention recovery changes the inputs used to fit MLP recovery. Starting from BF16 $M^{(0)}$, each intervention measures residual R_k on model state $M^{(k)}$, solves for a correction, and installs it:

$$R_k = \mathsf{Measure}_k(M^{(k)}), \qquad M^{(k+1)} = \mathsf{Apply}_k\big(M^{(k)}, \mathsf{Solve}_k(R_k)\big). \tag{11}$$

Each fit therefore uses the model state produced by the preceding intervention.

6 Experiments

We first evaluate HiRes across models and tasks and examine the contributions of its stages. We then test whether QuantForge improves algorithm discovery under matched evaluation budgets and isolate the role of residual compilation.

6.1 HiRes: Robust MXFP4 Quantization

Evaluation protocol. We evaluate Llama-3.2-3B (Meta, 2024), Qwen3-4B/8B/32B (Yang et al., 2025), Mistral-7B-v0.3 (Mistral AI, 2024), Llama-3-8B (Grattafiori et al., 2024), and OLMo-2-13B (Walsh et al., 2025). All use matched BF16 checkpoints, strict MXFP4 W4A4, and 128 ordered C4-train (Raffel et al., 2020) sequences of 2,048 tokens with seed 0; KV caching is disabled during evaluation. The seven tasks are WikiText-2 (Merity et al., 2017) and C4 perplexity, ARC-Challenge/ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2020) accuracy, normalized except on WinoGrande. Table 1 compares HiRes with nine baselines (protocols and endpoints in Appendix C.1). Discovery uses Qwen3-4B feedback and Llama-3.1-8B transfer validation (Appendix A.1). HiRes is frozen before evaluation on all other models, including Llama-3.2-3B and Mistral-7B-v0.3. Post-discovery ablations do not feed back into algorithm design.

Robust Fit penalizes methods whose good average masks large damage on one model. For task j on model m, BF16-relative damage is $d_{mj} = \mathrm{PPL}_q/\mathrm{PPL}_{16} - 1$ for perplexity and $d_{mj} = 1 - \mathrm{Acc}_q/\mathrm{Acc}_{16}$ for accuracy. We combine mean and worst-case damage first across tasks (Fit-7), then across models (Robust Fit):

$$\text{Fit-7}_m = \tfrac{1}{2}\,\mathrm{mean}_{j=1}^{7}\, d_{mj} + \tfrac{1}{2}\,\max_{j=1}^{7} d_{mj}, \tag{12}$$

$$\text{Robust Fit} = \tfrac{1}{2}\,\mathrm{mean}_{m=1}^{7}\,\text{Fit-7}_m + \tfrac{1}{2}\,\max_{m=1}^{7}\,\text{Fit-7}_m. \tag{13}$$

Lower is better. The same equal weighting applies to every method; Appendix C.2 reports alternative task summaries.

Cross-model robustness. HiRes achieves the lowest seven-model Robust Fit (0.09300), ahead of WUSH (0.09910) and BRQ (0.11152), and the lowest mean Fit-7 (0.06726). It ranks first on Mistral-7B and Llama-3-8B and improves Qwen3-8B Fit-7 from WUSH's 0.0500 to 0.0324. On the new OLMo-2 architecture and model family, the frozen algorithm achieves the lowest Fit-7

Table 1: Seven-model Fit-7 under the same MXFP4 operand/encoding contract (lower is better; best in bold).

Method	Llama 3B↓	Qwen 4B↓	Mistral 7B↓	Qwen 8B↓	Llama 8B↓	OLMo-2 13B↓	Qwen 32B↓	Mean↓	Robust↓
RTN	.2594	.2283	.1512	.1666	.2522	.1360	.0919	.18365	.22151
GPTQ	.1973	.1252	.1071	.1411	.2283	.1360	.0747	.14422	.18624
MR-GPTQ	.1387	.0986	.0541	.0587	.1394	.0846	.0524	.08951	.11447
BRQ	.1260	.0767	.0573	.0685	.1386	.0811	.0425	.08440	.11152
WUSH	.1224	**.0479**	.0503	.0500	.1248	.0823	.0365	.07345	.09910
TORQ	.2964	.1929	.1217	.1970	.2654	.1067	.0929	.18185	.23911
BATQuant+GPTQ	.2645	.2051	.1482	.1498	.2727	.1311	.1217	.18473	.22871
SpinQuant+GPTQ	.3191	.1409	.1142	.0835	.2588	.1317	.1023	.16435	.24171
FOCUS	.1803	.0738	.0749	.1095	.1737	.1195	.0671	.11411	.14718
HiRes	**.1174**	.0524	**.0452**	**.0324**	**.1187**	**.0773**	**.0274**	**.06726**	**.09300**

Table 2: Post-discovery Fit-3 ablations (WT2, C4, ARC-C; seed 0; lower is better). Top: cumulative stages. Bottom: interventions in full HiRes with other procedures unchanged (Appendices C.3–C.5).

Realized program	Qwen3-8B↓	Llama-3-8B↓
Strict RTN	.20249	.31653
+ Geometric shaping	.07828	.16741
+ Discrete realization	.06502	.15530
+ Structural recovery	**.03259**	**.14830**
Remove local interaction K	.04046	.15529
Remove diagonal balance ($D = I$)	.04326	.15921
Disable one-code refinement	.04397	.15359
Use stale MLP statistics	.04173	.15824

(0.077301), ahead of BRQ (0.081112) and WUSH (0.082295), with no model-specific tuning. Its WikiText-2/C4 perplexity is $5.658/11.339$, compared with WUSH's $5.699/11.376$. It also leads the Qwen3-32B comparison with Fit-7 0.027369, followed by WUSH (0.036489) and BRQ (0.042486; Table 9).

Hierarchical residual resolution. Each successive stage improves three-task fitness on both 8B models (Table 2). Geometry supplies the largest initial gain; discrete realization and structural recovery reduce the remaining damage.

Removing local interaction K, diagonal balance D, or one-code refinement increases Fit-3 on both models. Fitting MLP recovery before attention correction also increases damage at the same deployment order and solver budget, supporting remeasurement after the attention path changes. HiRes beats WUSH across three independent calibration seeds on both models. Component and recovery-path rankings also agree across seeds (Appendices C.3, C.4, and C.5).

6.2 QuantForge: Evidence-Guided Algorithm Discovery

Matched-budget protocol. We test whether controlled evidence and checked revisions improve discovery beyond scores, summaries, and reflection. Score-only, TextMem, ReflectMem, and QuantForge share the proposer, initial RTN program, editable interface, evaluator, and 240-call budget across eight paired seeds. Score-only uses code, scores, and execution feedback; TextMem adds free-form experiment summaries; ReflectMem adds reflection on successful and failed programs. QuantForge adds competing explanations, discriminating controls, and checked code responses. Search observes Qwen3-4B WikiText-2 perplexity. All arms evaluate held-out Llama-3.1-8B (Grattafiori et al., 2024) every 20 calls without returning results to search. QuantForge allocates 99 calls to controls and compliance probes and 141 to new programs; baselines use all 240 for new programs. Held-out probes are excluded from this budget. ReflectMem and QuantForge allow 65,536 input tokens, including at most 16,384 for memory (Appendix D.1).

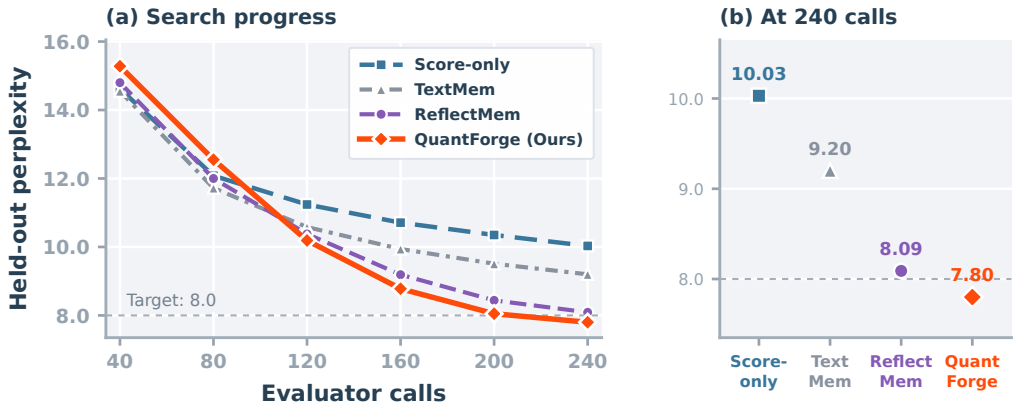

Figure 2: Held-out Llama-3.1-8B WT2 perplexity across eight paired seeds. (a) Recorded median search trajectories. (b) Final medians for all four arms at 240 calls, including ReflectMem. Search observes Qwen3-4B; dashed lines mark the 8.0 target.

Discovery efficiency. Despite evaluating only 141 new programs, QuantForge reaches median held-out PPL 7.80 at 240 calls, versus 8.09 for ReflectMem, 9.20 for TextMem, and 10.03 for Score-only. It initially progresses more slowly, then overtakes all three baselines between 80 and 120 calls (Figure 2). The PPL ≤ 8.00 target is reached in $6/8$, $3/8$, $3/8$, and $1/8$ runs, respectively. Against ReflectMem, QuantForge improves in $7/8$ paired seeds, with mean difference -0.250 and 95% CI $[-0.386, -0.114]$ (per-seed results and threshold crossings in Appendix D.1).

Value of residual compilation. Removing mandatory implementation and compliance checks tests whether compilation justifies its budget: the 30 released probe calls increase candidate evaluations from 141 to 171 within the same 240-call budget. With competing explanations, compiled controls, and memory fixed, median held-out PPL rises from 7.80 to 8.06. QuantForge wins $8/8$ paired seeds (mean difference -0.275; 95% CI $[-0.368, -0.182]$; Appendix D.2).

From verdict to code. Replaying 48 transitions from eight trajectories isolates how successors use the same evidence. Each pair shares the parent program, history, and control result. Requiring verdict implementation rather than free reflection increases improving successors from $15/48$ (31%) to $28/48$ (58%) and improves mean visible PPL change from -0.04 to -0.21 (Table 38), with more improving successors in all eight trajectories.

Control-selection and implementation-checking ablations appear in Appendix D.3; Table 34 reports which stages of the hierarchy in Section 5 appear in retained programs.

7 Conclusion

We introduced QuantForge to discover PTQ algorithms through progressive residual factorization. Controlled experiments resolve design questions and guide subsequent code changes; remeasurement identifies what remains. This process produced HiRes, whose quantization stages operate on successively realized model states. Its cross-model aggregate and 32B transfer results support the discovered construction.

Matched-budget experiments also show that QuantForge improves held-out performance while spending fewer evaluations on new candidates. Stagewise and fresh-residual controls support the dependence between successive interventions. Experiments can therefore guide how an algorithm is decomposed as well as which program is retained. The primary limitation of our evaluation is that realized latency, throughput, and memory remain untested on native MXFP4 kernels. We leave this deployment study to future work.

AI USE STATEMENT

The authors designed the QuantForge framework and directed the research. Generative AI tools supported iterative algorithm development within the framework and assisted with manuscript language editing and proofreading. The authors reviewed and verified the technical content, experiments, analyses, and conclusions.

REPRODUCIBILITY STATEMENT

Appendix A specifies QuantForge's search protocols, controlled experiments, and program-update procedure. Appendix B details HiRes's geometric construction, discrete quantization rules, and structural recovery. Appendix C documents the models, calibration and evaluation protocols, repeated runs, transfer evaluations, and algorithm ablations. Appendix D reports the matched-budget framework comparisons, proposer settings, and controlled ablations supporting the empirical results.

REFERENCES

Dhruv Agarwal, Bodhisattwa Prasad Majumder, Reece Adamson, Megha Chakravorty, Satvika Reddy Gavireddy, Aditya Parashar, Harshit Surana, Bhavana Dalvi Mishra, Andrew McCallum, Ashish Sabharwal, and Peter Clark. AutoDiscovery: Open-ended scientific discovery via Bayesian surprise. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/23b127521af7ca7a42f5cdb7507be4f2-Abstract-Conference.html.

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://aclanthology.org/2023.emnlp-main.298/.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 100213–100240, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b5b939436789f76f08b9d0da5e81af7c-Abstract-Conference.html.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, et al. Lessons from the trenches on reproducible evaluation of language models, 2024. URL https://arxiv.org/abs/2405.14782.

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

Jiale Chen, Vage Egiazarian, Roberto L. Castro, Torsten Hoefler, and Dan Alistarh. WUSH: Near-optimal adaptive transforms for LLM quantization. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ZsECxUkbKB.

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, 2018. URL https://arxiv.org/abs/1803.05457.

David L. Donoho and Iain M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994. doi: 10.1093/biomet/81.3.425.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015. doi: 10.1126/science.aaa9375.

Vage Egiazarian, Roberto Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Bridging the gap between promise and performance for microscaling FP4 quantization. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/b87bb4f6346d727b265088235e5bc389-Abstract-Conference.html.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Juraj Gottweis et al. Towards an AI co-scientist, 2025. URL https://arxiv.org/abs/2502.18864v1. Preprint, version 1.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634.

Kexin Huang, Ying Jin, Ryan Li, Michael Y. Li, Emmanuel Candès, and Jure Leskovec. Automated hypothesis validation with agentic sequential falsifications. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 25372–25437, 2025. URL https://proceedings.mlr.press/v267/huang25n.html.

Ji-Fu Li, Manyi Zhang, Xiaobo Xia, Han Bao, Haoli Bai, Zhenhua Dong, and Xianzhi Yu. BATQuant: Outlier-resilient MXFP4 quantization via learnable block-wise optimization. In *European Conference on Computer Vision*, 2026a. URL https://eccv.ecva.net/virtual/2026/poster/4636.

Zhongzheng Li, Tiancan Feng, Wenhao Li, et al. OPTScientist: Multi-agent discovery of typed optimizer programs for transformer pretraining, 2026b. URL https://arxiv.org/abs/2607.20486. Preprint.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. DuQuant: Distributing outliers via dual transformation makes stronger quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 87766–87800, 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/9febda1c8344cc5f2d51713964864e93-Abstract-Conference.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, 2024b. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32201–32223, 2024. URL https://proceedings.mlr.press/v235/liu24bs.html.

Qi Liu, Ruochen Hao, Can Li, and Wanjing Ma. OR-Agent: Bridging evolutionary search and structured research for automated algorithm discovery, 2026. URL https://arxiv.org/abs/2602.13769v3. Preprint, version 3.

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://proceedings.iclr.cc/paper_files/paper/2025/hash/e5b1c0d4866f72393c522c8a00eed4eb-Abstract-Conference.html.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292. Preprint.

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.

Meta. Llama-3.2-3B model card, 2024. URL https://huggingface.co/meta-llama/Llama-3.2-3B.

Mistral AI. Mistral-7B-v0.3 model card, 2024. URL https://huggingface.co/mistralai/Mistral-7B-v0.3.

Youssef Mroueh, Carlos Fonseca, Brian Belgodere, and David Cox. CliffSearch: Structured agentic co-evolution over theory and code for scientific algorithm discovery, 2026. URL https://arxiv.org/abs/2604.01210. Preprint.

Alexander Novikov et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131. Preprint.

Adam Paszke, Sam Gross, Francisco Massa, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://www.jmlr.org/papers/v21/20-074.html.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Bita Darvish Rouhani, Nitin Garegrat, Tom Savell, Ankit More, Kyung-Nam Han, Ritchie Zhao, Mathew Hall, Jasmine Klar, Eric Chung, Yuan Yu, Michael Schulte, Ralph Wittig, Ian Bratt, Nigel Stephens, Jelena Milanovic, John Brothers, Pradeep Dubey, Marius Cornea, Alexander Heinecke, Andres Rodriguez, Martin Langhammer, Summer Deng, Maxim Naumov, Paulius Micikevicius, Michael Siu, and Colin Verrilli. OCP microscaling formats (MX) specification. Specification Version 1.0, Open Compute Project, 2023a. URL https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning, 2023b. URL https://arxiv.org/abs/2310.10537. Preprint.

Peter J. Rousseeuw and Christophe Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424):1273–1283, 1993. doi: 10.1080/01621459.1993.10476408.

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. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Yuantian Shao, Peisong Wang, Yuanteng Chen, Chang Xu, Zhihui Wei, and Jian Cheng. Block rotation is all you need for MXFP4 quantization. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=cyR2FtZzY1.

Noam Shazeer. GLU variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Niki van Stein and Thomas Bäck. LLaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 29(2): 331–345, 2025. doi: 10.1109/TEVC.2024.3497793.

Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, et al. 2 OLMo 2 furious (COLM's version). In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=2ezugTT9kU.

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, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

Zukang Xu, Xing Hu, and Dawei Yang. TORQ: Two-level orthogonal rotation for MXFP4 quantization, 2026. URL https://arxiv.org/abs/2605.19561. Preprint.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, et al. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search, 2025. URL https://arxiv.org/abs/2504.08066. Preprint.

Xianglong Yan, Hong Liu, Chengzhu Bao, Tianao Zhang, Guanghua Yu, Jianchen Zhu, and Yulun Zhang. FOCUS: FP4 optimization via coupled-relaxation and dual-granularity scaling. *arXiv preprint arXiv:2608.01847*, 2026. URL https://arxiv.org/abs/2608.01847.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Haoran Ye, Jiarui Wang, Zhiguang Cao, et al. ReEvo: Large language models as hyper-heuristics with reflective evolution. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/4ced59d480e07d290b6f29fc8798f195-Abstract-Conference.html.

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, 2019. URL https://aclanthology.org/P19-1472/.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. Monte Carlo tree search for comprehensive exploration in LLM-based automatic heuristic design. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78338–78373, 2025. URL https://proceedings.mlr.press/v267/zheng25o.html.

APPENDIX

A QuantForge: SEARCH PROCEDURE

We first distinguish the two discovery protocols, then describe how QuantForge turns a controlled experiment into a program update.

A.1 DISCOVERY PROTOCOLS

Algorithm discovery. Algorithm development starts from GPTQ (Frantar et al., 2023) and retains successful descendants as starting points for later revisions. The search can revise the transform, discrete reconstruction, and residual recovery; these components are not fixed to the initial implementation. Each candidate is rebuilt from BF16 weights using the same strict-MXFP4 contract and 128×2048 calibration protocol.

Qwen3-4B (Yang et al., 2025) is the search-visible model. Its task results and diagnostic measurements guide subsequent proposals. Llama-3.1-8B (Grattafiori et al., 2024) serves as the transfer-validation model. A visible-model improvement is accepted as transferable only when this validation model also improves against the same comparison program. The proposer receives the aggregate transfer acceptance signal; detailed task, layer, and module diagnostics remain with the evaluator.

Llama-3.2-3B, Mistral-7B-v0.3, Qwen3-8B, and Meta-Llama-3-8B are excluded from proposal generation and candidate selection and are evaluated only after the final algorithm is fixed. Component and stage ablations on Qwen3-8B and Meta-Llama-3-8B then measure the contribution of its parts. Their results do not feed back into discovery or component selection.

The search score combines WikiText-2 and C4 perplexity with ARC-Challenge length-normalized accuracy (`acc_norm`) (Merity et al., 2017; Raffel et al., 2020; Clark et al., 2018). For each model m, damage is measured against its own BF16 reference:

$$d_{m,\mathrm{WT2}} = \frac{\mathrm{PPL}_{q,m,\mathrm{WT2}}}{\mathrm{PPL}_{16,m,\mathrm{WT2}}} - 1, \qquad d_{m,\mathrm{C4}} = \frac{\mathrm{PPL}_{q,m,\mathrm{C4}}}{\mathrm{PPL}_{16,m,\mathrm{C4}}} - 1, \tag{14}$$

$$d_{m,\mathrm{ARC}} = \frac{\mathrm{Acc}^{\mathrm{norm}}_{16,m} - \mathrm{Acc}^{\mathrm{norm}}_{q,m}}{\mathrm{Acc}^{\mathrm{norm}}_{16,m}}, \tag{15}$$

$$\text{Fit-3}_m = \frac{d_{m,\mathrm{WT2}} + d_{m,\mathrm{C4}} + d_{m,\mathrm{ARC}}}{6} + \frac{1}{2} \max_{j \in \{\mathrm{WT2,C4,ARC}\}} d_{mj}. \tag{16}$$

Lower is better. Equal weight on mean and worst-task damage rewards broad improvements while penalizing a large loss on any one task. Damage is not clipped at zero, so gains over BF16 retain their negative sign.

Independent framework comparison. The matched-budget study in Section 6.2 starts each arm from the same strict-MXFP4 round-to-nearest (RTN) program. The proposer revises this code using Qwen3-4B scores and execution feedback; QuantForge additionally carries forward unresolved questions and controlled experimental conclusions. Each accepted revision supplies the parent state for later interventions. The editable program can introduce new operations, so the final three-stage quantizer is not part of the initial RTN program.

Qwen3-4B is search-visible; Llama-3.1-8B is held out for independent transfer assessment. Every 20 evaluator calls, a separate evaluator tests the current champion on Llama. Neither its score nor its per-task, per-layer, or per-module diagnostics are returned to the proposer or used to choose descendants. This evaluator separation follows the holdout principle (Dwork et al., 2015); the recorded outcomes measure transfer under the same search budget.

This controlled study uses WikiText-2 perplexity as its visible objective and held-out endpoint. Its $\mathrm{PPL} \leq 8$ success gate and 240-call results describe this independent framework comparison. The algorithm-development search above uses Fit-3; the quantizer benchmark uses the seven-task panel.

A.2 Controlled Experiments and Program Updates

Each proposal specifies an executable candidate p, its parent π, and the residual ρ it aims to reduce. It also records a focal explanation and its competitors in $\mathcal{E}$, their predicted outcomes $P_e(c)$ under each condition c, and the allowed controls $\mathcal{U}$.

Before execution, the controller applies Eq. 2 and freezes the selected controls together with p, π, ρ, $\mathcal{E}$, P, and the evaluator identity. The primary run and its controls share this packet. A completed condition can be reused only when the full execution identity matches. If the allowed controls cannot distinguish the registered explanations, the proposal is revised before model evaluation. Every condition shares the model, calibration data, quantizer format, evaluator, and resource ceiling; only the declared intervention changes.

The observed results determine both the status of the explanation and the required response in the next program:

Verdict	Residual status	Required program response
Supported	closed under the focal account	extend or compose the direction
Target-only	gain lacks the predicted signature	repair the explanation
Mediator-only	signature lacks the target gain	repair transfer to the target
No effect	intervention is parent-equivalent	retire the intervention
Unresolved	current controls do not separate accounts	add a discriminator
Unsafe	rollback or contract check fails	reject and restore the parent

Program retention is evaluated separately. A target-only candidate, for example, may remain useful as executable code even though its focal account cannot guide a descendant.

For every response-bearing descendant, the controller compares the declared response with the changed source and an executed probe. A source-level no-op does not count as a residual closure. The check is scoped to the computation named by ρ; it does not require unrelated code to change.

A.3 Search Cycle

The search maintains executable programs and unresolved residuals as separate frontiers. The controller selects parents and schedules experiments; the residual compiler determines which conclusions can guide a program update. Population size, mutation operators, and scheduling can vary while retaining this interface.

After compilation, the revised program is remeasured and exposes the next intervention set:

$$\mathcal{R}(P_{t+1}) = \mathsf{Measure}(P_{t+1}), \qquad \mathcal{S}_{t+1} = \mathsf{Open}(P_{t+1}, \mathcal{R}(P_{t+1})). \tag{17}$$

Here $\mathcal{R}(P_{t+1})$ denotes the residuals remeasured on this program, not a replacement of the full residual frontier. Unresolved questions attached to other retained programs remain available. $\mathcal{S}_{t+1}$ contains interventions made actionable by residuals measured on P_{t+1}. It can include existing operations or newly introduced components. The hierarchy records both the operation and the earlier program state that made it relevant.

The history stores completed experiments, Pareto donor programs, duplicate proposals, and stagnation information alongside the two frontiers. These records let the proposer reuse successful computations and distinguish a new condition from a repeated intervention.

Algorithm 2 One QuantForge round

1: **input:** executable frontier $\mathcal{P}_t$, residual frontier $\mathcal{R}_t$, frozen contract $\mathcal{C}$
2: select a program and an actionable residual
3: propose executable interventions with competing explanations
4: compile minimum-cost discriminating controls (Eq. 2)
5: freeze and execute the experiment packet under $\mathcal{C}$
6: retain programs by measured performance
7: resolve the residual and verify the required source-level response
8: remeasure residuals on the changed program; open the next intervention space

B HiRes: Algorithm Details

The details below follow the three stages in Section 5. Each stage uses the model state produced by the preceding one; the table below identifies the measurements and outputs at these boundaries.

B.1 Quantization Setup

The complete implementation starts from fresh BF16 weights and the frozen exact 128×2048-token calibration sequence. It targets the decoder query, key, value, output, gate, up, and down projections. Both operands use contiguous 32-value blocks, E2M1 element values, and E8M0 shared exponents. The representation follows the MX specification (Rouhani et al., 2023a;b); our deterministic encoder conventions are given below. Weights form a static decoded W4 representation; activations are quantized dynamically to A4. Embeddings, normalization, RoPE (Su et al., 2024), and the language-model head remain at higher precision. KV caching is disabled during evaluation for every model, including Qwen3-32B (`use_cache=False`). Curvature uses pre-A4 calibration activations. Evaluation uses logical fake quantization in PyTorch, with explicit checks of element values and shared scales, to measure numerical quality.

For a nonzero block x, the reference encoder sets $e = \mathrm{clip}(\lfloor \log_2 \max_i |x_i| \rfloor - 2, -127, 127)$ and scale 2^e; an all-zero block uses $e = -127$. Element magnitudes are drawn from $\{0, \frac{1}{2}, 1, \frac{3}{2}, 2, 3, 4, 6\}$. Nearest-code ties select the larger magnitude; the sign bit follows the input sign. These encoder conventions are fixed across our comparisons.

Transform statistics and pair solves use FP64; transforms and their inverses use FP32 during construction. The optimized inference layout retains the folded transform $U = DT$ and releases the construction-only inverse; it stores G_Q in upper-triangular form and K_Q in full FP32 (Appendix C.6). GPTQ uses damping ratio 0.01 and FP32 reconstruction, with decoded weights stored in BF16. Activation-code scores use FP32. Structural fits accumulate in FP64; their runtime corrections use FP32 arithmetic followed by BF16 rounding. The frozen implementation stages captured activation arrays in FP16 between calibration passes. Strict W4A4 applies to all target linear-layer operands. The low-dimensional recovery arithmetic runs outside the quantized GEMM and introduces no higher-precision weight matrix.

Stage	Measured state	Output
Geometric shaping	BF16 weights, calibration inputs, and anchor quantization errors	Transform T and balance D
Discrete realization	Installed DT coordinates and transformed pre-A4 inputs	W4 weights and dynamic A4 rule
Structural recovery	Installed W4/A4 operators, then the attention-corrected state	Attention and MLP coefficients

Construction starts from BF16 weights. We remeasure transformed activations after shaping, fit attention recovery on the encoded operators, and remeasure the resulting state before fitting MLP recovery. This is the conditional measurement order in Eq. 11.

B.2 Geometric Shaping

Reference coordinates. For a linear operator $Y = XW^\top$, Stage I applies an invertible map T by $\widetilde{X} = XT^\top$ and $\widetilde{W} = WT^{-1}$. Therefore $\widetilde{X}\widetilde{W}^\top = XW^\top$, and the BF16 function is unchanged before discretization. Diagonal balancing uses the same paired folding rule. The map $T = R_H G P_{\mathrm{ref}}$ combines reference coordinates P_{ref}, a local interaction correction G, and the normalized Hadamard R_H. All quantities below are computed independently for each physical Block32 group. With $q = 32$, the ridge coefficients in Eq. 4 are

$$\lambda_A = .01\ \mathrm{tr}(X^\top X/n)/q + 2^{-30}, \qquad \lambda_B = .01\ \mathrm{tr}(W^\top W/m)/q + 2^{-30}. \tag{18}$$

Determinant normalization is $\mathrm{DN}(M) = M/\det(M)^{1/q}$. For the spectral cap in Eq. 4, write $S = U\,\mathrm{Diag}(\lambda)U^\top$,

$$\mathrm{Cap}_\kappa(S) = U\,\mathrm{Diag}\Big[\exp\Big(\Pi_{\mathbf{1}^\top z=0,\,\|z\|_\infty \le \frac{1}{2}\log\kappa}(\log\lambda)\Big)\Big]\,U^\top. \tag{19}$$

Here S is positive definite, $\log \lambda$ is elementwise, and $\Pi_{\mathcal{C}}$ denotes Euclidean projection onto $\mathcal{C}$. For the zero-sum box, the projected entries are $z_i = \mathrm{clip}(x_i - \tau, -b, b)$, with $x_i = \log \lambda_i$ and $b = \frac{1}{2} \log \kappa$; τ makes $\sum_i z_i = 0$. The capped eigenvalues have product one and ratio at most κ. The implementation uses a sorted-breakpoint projection for this cap.

Local interaction correction. One anchor quantization identifies errors left by the reference coordinates. The following construction turns those errors into a bounded interaction matrix K, which determines G in Eq. 6. The anchor uses $Y = XP_{\mathrm{ref}}^{\top}$ and $V = WP_{\mathrm{ref}}^{-1}$ before the normalized Hadamard $R_H = H_{32}/\sqrt{32}$. Let Q_A, Q_W be the strict anchor's activation and weight quantizers. Their errors, mapped back through R_H, are

$$E_X = [Q_A(YR_H^{\top}) - YR_H^{\top}]R_H, \qquad E_W = [Q_W(VR_H^{\top}) - VR_H^{\top}]R_H. \tag{20}$$

The full anchor statistics are

$$C = Y^{\top}Y/n, \quad C_W = V^{\top}V/m, \quad K_X = E_X^{\top}Y/n, \quad K_W = V^{\top}E_W/m. \tag{21}$$

Activations are split by calibration-sequence parity, while weight rows are split by row parity. Let $f \in \{0, 1\}$ index these folds, and let $C^{(f)}, C_W^{(f)}, K_X^{(f)}$, and $K_W^{(f)}$ denote the fold-specific versions of Eq. 21. Write $c_i^{(f)} = C_{ii}^{(f)}$, $h_i^{(f)} = [C_W^{(f)}]_{ii}$, and

$$a_{ij}^{(f)} = h_i^{(f)}[K_X^{(f)}]_{ij}, \quad b_{ij}^{(f)} = -c_j^{(f)}[K_W^{(f)}]_{ij}, \quad p_{ij}^{(f)} = a_{ij}^{(f)} + b_{ij}^{(f)}. \tag{22}$$

The curvature proxy is

$$q_{ij}^{(f)} = (2 + 2^{-8})h_i^{(f)}c_j^{(f)} + 2^{-30}\bar{h}^{(f)}\bar{c}^{(f)}, \tag{23}$$

where bars denote the mean diagonal value.

For each unordered pair $i < j$, define reciprocal and directed coordinates

$$\mathcal{M}(z_{ij}, z_{ji}) = \left(\frac{z_{ij} + z_{ji}}{\sqrt{2}}, \frac{z_{ij} - z_{ji}}{\sqrt{2}}\right). \tag{24}$$

Let $\bar{p}$, $\bar{a}$, and $\bar{b}$ be the averages of their transformed coordinates over the two folds. The threshold combines

$$\delta_{\mathrm{fold}} = \tfrac{1}{2}|\mathcal{M}(p^{(0)}) - \mathcal{M}(p^{(1)})|, \tag{25}$$

$$\delta_{\mathrm{own}} = \tfrac{1}{2}\big(|\bar{a}| + |\bar{b}| - |\bar{a} + \bar{b}|\big), \tag{26}$$

$$\delta_{\mathrm{carrier}} = \left(0,\ [|\bar{p}_{\mathrm{dir}}| - |\bar{p}_{\mathrm{rec}}|]_+\right), \qquad \delta = \delta_{\mathrm{fold}} + \delta_{\mathrm{own}} + \delta_{\mathrm{carrier}}. \tag{27}$$

The first term rejects fold-specific effects. The second penalizes disagreement between weight and activation attribution. The last caps the surviving directed magnitude by the reciprocal magnitude before the other two penalties are applied.

Let $\bar{q}$ be the mean of Eq. 23 over both folds and all off-diagonal entries, and define

$$\widehat{q}_{ij} = 1 + \frac{q_{ij}^{(0)} + q_{ji}^{(0)} + q_{ij}^{(1)} + q_{ji}^{(1)}}{4(\bar{q} + 2^{-30})}. \tag{28}$$

For both pair modes, the unprojected coefficient is

$$z_{ij} = \mathrm{Soft}\left(\frac{\bar{p}_{ij}}{(\bar{q} + 2^{-30})\sqrt{\widehat{q}_{ij}}}, \frac{\delta_{ij}}{(\bar{q} + 2^{-30})\sqrt{\widehat{q}_{ij}}}\right). \tag{29}$$

Here z_{ij} is a two-vector (reciprocal and directed), all absolute values and thresholds act componentwise, and $[x]_+ = \max(x, 0)$. The soft-threshold operator is $\mathrm{Soft}(x, t) = \mathrm{sign}(x)[|x| - t]_+$, the standard shrinkage rule (Donoho & Johnstone, 1994). All $2\binom{32}{2}$ coefficients in a block are jointly projected onto the Euclidean ball of radius $1/8$, producing $(v_{ij}^{\mathrm{rec}}, v_{ij}^{\mathrm{dir}})$. The dense update is reconstructed as

$$K_{ij} = \frac{v_{ij}^{\mathrm{rec}} + v_{ij}^{\mathrm{dir}}}{\sqrt{2\widehat{q}_{ij}}}, \qquad K_{ji} = \frac{v_{ij}^{\mathrm{rec}} - v_{ij}^{\mathrm{dir}}}{\sqrt{2\widehat{q}_{ij}}}, \qquad K_{ii} = 0. \tag{30}$$

Equations 22–30 give K in one pass from the anchor statistics. Since $\widehat{q}_{ij} \geq 1$, this construction gives $\|K\|_2 \leq \|K\|_F \leq \|v\|_2 \leq 1/8$. $I + K$ is therefore invertible with positive determinant, justifying the volume normalization in Eq. 6.

Weight–activation balance. After fixing T, diagonal scaling D redistributes quantization risk between the two operands. The eight four-coordinate subgroups in each Block32 group use the risks in Eq. 7. For subgroup activation and weight matrices X_g, W_g, define $C_X = X_g^\top X_g/n$, $C_E = E_X^\top E_X/n$, $G_W = W_g^\top W_g$, and $G_E = E_W^\top E_W$, with errors measured after strict encoding. These are uncentered second moments, not centered covariances. For physical block b, define $\epsilon_b = 2^{-12}\,\mathrm{median}_g(r_{A,bg} + r_{W,bg})$. For $r_{A,bg} + \epsilon_b > 0$ and $r_{W,bg} + \epsilon_b > 0$ in every subgroup, the proposed log-scale vector is

$$u_b = \Pi_{\mathbf{1}^\top u=0,\,\|u\|_\infty \le 1/8}\left[\frac{1}{8}\left(\log_2 \frac{r_{A,b}+\epsilon_b}{r_{W,b}+\epsilon_b} - \mathrm{mean}_g \log_2 \frac{r_{A,bg}+\epsilon_b}{r_{W,bg}+\epsilon_b}\right)\right]. \tag{31}$$

The zero-sum box projection uses 80 scalar bisection steps. Each real-valued component of $u_b \in \mathbb{R}^8$ is broadcast to four coordinates in D; the later E8M0 encoding uses integer shared exponents. Each block tests u_b, $u_b/2$, and zero by re-encoding both operands. It accepts the full step if its W4/A4 reconstruction loss does not exceed the identity loss, otherwise the half step if it passes, and otherwise the identity. For proposal s, this loss is

$$J_b(s) = \frac{\|\widehat{X}_b(s)\widehat{W}_b(s)^\top - X_b W_b^\top\|_F^2}{\max(\|X_b W_b^\top\|_F^2, \varepsilon_{64})}, \qquad \varepsilon_{64} = 2^{-1022}. \tag{32}$$

Here X_b, W_b are the Stage-I transformed operands before rescaling. Each trial recomputes GPTQ and activation refinement; the final mixed full/half/identity choice is then re-encoded once more.

B.3 Discrete Realization

GPTQ (Frantar et al., 2023) reconstructs weights in the fixed DT coordinates using curvature from the transformed pre-A4 inputs. Once these W4 weights are installed, activation refinement accounts for their encoding error when choosing an A4 code.

The score in Eq. 8 follows by expanding the block-local output error. For column vectors $x, q \in \mathbb{R}^{32}$, let $r_0 = \widehat{W}(q_0 - x) - (W - \widehat{W})x$. Replacing q_0 by $q_0 + \delta e_i$ changes $\|r_0\|_2^2$ by $2\delta[\widehat{W}^\top r_0]_i + \delta^2[\widehat{W}^\top\widehat{W}]_{ii}$, which is Eq. 8. Thus the score is the exact change in the output-error objective for this block.

The implementation enumerates all 16 E2M1 bit patterns, including the two signed zeros, for each of the 32 coordinates at the fixed block exponent. It accepts the pair (i, c) with the smallest score only if that score is negative. A zero tie retains nearest rounding; negative-score ties choose the lowest coordinate index and then the lowest unsigned code index. At most one element code changes per physical block.

B.4 Structural Recovery

Recovery fits the residuals of the installed W4/A4 model, first along the attention path and then through the gated MLP.

Attention recovery. The attention fit in Eq. 9 pools tokens and head coordinates separately for each layer and key–value head. Writing $v_{t,j}$ for coordinate j of $\mathbf{v}_t$, the sums S_x, S_y, S_{xx}, S_{xy} accumulate $v_{t,j}$, $v^\star_{t,j}$, $v^2_{t,j}$, and $v_{t,j}v^\star_{t,j}$, respectively, and N counts the pooled scalar pairs. The solution is

$$a = \frac{S_{xy} - S_x S_y/N}{S_{xx} - S_x^2/N}, \qquad b = \frac{S_y - aS_x}{N}. \tag{33}$$

When the denominator is zero, the implementation uses $(a, b) = (1, 0)$. The offset b is broadcast across the head coordinates. The affine map is evaluated in FP32, then rounded to BF16 before continuing through attention.

MLP recovery. After installing the attention correction, we collect new calibration outputs for the gated-MLP fit. It uses a first-order model of $m = \mathrm{SiLU}(g) \odot u$. Let $s = \mathrm{sigmoid}(g)$, $\phi' = s + gs(1-s) = \mathrm{SiLU}'(g)$, and $e = m^\star - m$. The regularized standard deviation uses an expectation over all calibration tokens and hidden coordinates in one layer:

$$\sigma_g = \sqrt{[\mathbb{E}[g^2] - \mathbb{E}[g]^2]_+ + 2^{-12}\mathbb{E}[g^2] + 2^{-20}}. \tag{34}$$

The local design vector is

$$D_{tj} = [\,\sigma_g \phi'_{tj} u_{tj},\ g_{tj} \phi'_{tj} u_{tj},\ m_{tj}\,]. \tag{35}$$

Each $D_{tj} \in \mathbb{R}^{1\times 3}$ is a row vector, and the layer shares one coefficient vector $\theta \in \mathbb{R}^3$. The BF16 down-projection energy assigns coordinate weight

$$\widetilde{\omega}_j = \mathrm{clip}\left(\frac{\|W_{\mathrm{down}}[:,j]\|_2^2}{\mathrm{mean}_k \|W_{\mathrm{down}}[:,k]\|_2^2 + 2^{-20}}, \frac{1}{4}, 4\right), \qquad \omega_j = \frac{\widetilde{\omega}_j}{\mathrm{mean}_k\, \widetilde{\omega}_k}. \tag{36}$$

Calibration is partitioned into 32 cells: eight sequence residue classes and four 512-token quarters. In cell c, the local fit is

$$\theta_c = (G_c + \lambda_c I)^{-1} z_c, \qquad G_c = \mathbb{E}_{(t,j)\in c}[\omega_j D_{tj}^\top D_{tj}], \quad z_c = \mathbb{E}_{(t,j)\in c}[\omega_j D_{tj}^\top e_{tj}], \tag{37}$$

This is the weighted ridge solution (Hoerl & Kennard, 1970), with $\lambda_c = 2^{-8}\,\mathrm{tr}(G_c)/3 + 2^{-20}\mathbb{E}_c[\omega m^2] + 2^{-30}$. For each of the three coordinates, let μ be the median of the 32 cell estimates and $d = 1.4826\,\mathrm{median}_c\,|\theta_c - \mu|$, the normal-consistent median absolute deviation (Rousseeuw & Croux, 1993). Both medians select the lower middle order statistic, matching the frozen implementation. The aggregate is

$$\widetilde{\theta} = \mathrm{sign}(\mu)[|\mu| - d/\sqrt{32}]_+, \qquad \theta = \widetilde{\theta}\,\min\left(1, \frac{1/8}{\|\widetilde{\theta}\|_1 + 2^{-20}}\right). \tag{38}$$

Writing $\theta = (\alpha, \beta, \eta)$ yields the update in Eq. 10. The BF16 corrected intermediate is then quantized by the ordinary dynamic-A4 pre-hook of the down projection.

C HiRes: Additional Results

The results cover model accuracy, evaluation stability, the role of each stage, and computational and storage costs. All ablations examine the frozen algorithm.

Question	Comparison	Section
Accuracy across models	Seven-model endpoints from 3B to 32B	C.1
Sensitivity to evaluation	Calibration seeds and score aggregation	C.2
Role of geometry	Stage progression, K/D ablations, and geometry replacement	C.3
Role of code refinement	Rounding rules, local errors, and runtime cost	C.4
Role of recovery paths	Attention, MLP, and their combination	C.5
Role of remeasurement	Pre/post-encoding and fresh/stale statistics	C.5
Computational and storage costs	Persistent storage, full-forward latency, and quantization time	C.6

C.1 Complete Quantization Results

All methods use the same BF16 checkpoints and exact 128×2048 C4-train calibration with seed 0 (Raffel et al., 2020). Tokenizer, token IDs, sample order, and caches are frozen per model, since evaluation conventions can affect comparisons (Biderman et al., 2024). The endpoints are WikiText-2 (Merity et al., 2017) and C4 perplexity, normalized accuracy on ARC-Challenge/ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and PIQA (Bisk et al., 2020), and accuracy on WinoGrande (Sakaguchi et al., 2020). ARC-Challenge uses all 299 official validation questions in their original order, read from `arc_challenge/validation.parquet`. The baseline suite comprises RTN, GPTQ, MR-GPTQ, BRQ, WUSH, TORQ, BATQuant+GPTQ, SpinQuant+GPTQ, and FOCUS. These results compare the methods under our common MXFP4 operand/encoding contract.

MR-GPTQ. We adapt the official FP-Quant implementation (Egiazarian et al., 2026), pinned at `239c4168`. It retains its transform and intrinsic GPTQ reconstruction, with encoding and decoding adapted to our E2M1/E8M0 contract. For OLMo, the loader divides the decoded exported scale by four to recover the intended scale; the codes and calibration-built artifact are unchanged. We reevaluate all seven tasks with this loader.

WUSH. We adapt the official implementation (Chen et al., 2026), pinned at `0b69f66c`, preserving its transform and intrinsic GPTQ reconstruction. Encoding and decoding follow our E2M1/E8M0 contract.

BRQ. BRQ (Shao et al., 2026) uses an official-code direct-core adaptation (`0e4ab0b9`): normalized Sylvester H_{32} transforms on consecutive input-feature blocks, followed by strict GPTQ. This port fixes the block transform; the paper's stochastic-sign, sharing, and RNG choices remain unresolved. The short table label includes the GPTQ reconstruction step.

TORQ. TORQ (Xu et al., 2026) uses our independent reimplementation of its two-level inter/intra-block rotation, followed by strict GPTQ. The short table label includes this reconstruction step.

BATQuant+GPTQ. Our independent BATQuant reimplementation (Li et al., 2026a) learns GPK transforms and clipping parameters for 160 optimizer steps per transformation group, then applies strict GPTQ. It uses the common C4 calibration cache in place of the paper's self-generated calibration data. GPTQ reconstructs the transformed weights once, with damping 0.01; all seven tasks restore the resulting payload with dynamic A4.

SpinQuant+GPTQ. We use the Brevitas implementation of learned SpinQuant rotations (Liu et al., 2025), pinned at `6ba27e0b`, fuse them into the float model, and apply one strict GPTQ pass. On OLMo2's post-norm topology, the port uses paired V/O Hadamard rotations across attention, optimized for 100 steps with global batch size eight. The fused float model preserves Base logits to relative RMS error 5.29×10^{-7}; GPTQ then reconstructs all 280 target projections. The OLMo row reports this topology-safe paired adaptation, rather than the original Llama residual-rotation recipe. For these compositions, `+GPTQ` denotes terminal weight reconstruction, not a second quantization of an already-built payload.

FOCUS. We use the official AngelSlim implementation of FOCUS (Yan et al., 2026), with coupled-relaxation scaling and four 8-value subgroups per physical Block32. Optimization keeps BF16 weights frozen and learns scales for one epoch with global batch size 32. Scale and subgroup-relaxation learning rates are 0.02 and 0.05, respectively; the distillation loss uses the top 1,000 teacher logits. Activation quantization remains dynamic MXFP4 without learned activation scales. Export retains FP4 codes and E8M0 dequantization scales, discarding subgroup-relaxation parameters. We replace the public example's calibration data with our frozen C4 128×2048 token cache and adapt module registration for the architectures beyond its Qwen3-4B example. The quantization algorithm is unchanged. All seven tasks evaluate the same exported artifact per model. FOCUS improves on GPTQ's Fit-7 across all seven models, with Robust Fit 0.14718, compared with 0.18624 for GPTQ and 0.09300 for HiRes.

Tables 3–9 report every raw endpoint used in the seven-model comparison. Accuracy values are percentages, except for the fractions in Table 9. On Llama-3-8B, HiRes lowers WikiText-2/C4 perplexity from WUSH's $7.133/11.117$ to $7.111/11.026$; on Mistral-7B, the corresponding values fall from $5.689/8.834$ to $5.664/8.810$.

Table 3: Complete seven-task results on Llama-3.2-3B Base.

Method	WT2↓	C4↓	ARC-C↑	ARC-E↑	HS↑	PIQA↑	Wino↑	Fit-7↓
BF16	7.818	11.216	41.47	70.50	72.79	77.58	61.17	.0000
RTN	10.266	15.189	35.79	61.83	65.94	72.42	57.30	.2594
GPTQ	9.768	14.177	37.12	62.71	67.04	74.10	57.46	.1973
MR-GPTQ	9.011	13.333	36.45	67.30	68.67	76.61	58.48	.1387
BRQ	8.969	13.182	39.46	67.63	68.39	75.90	58.56	.1260
WUSH	8.905	13.056	37.46	65.87	69.14	76.28	59.19	.1224
TORQ	10.558	15.597	31.77	57.87	63.94	71.55	57.46	.2964
BATQuant+GPTQ	10.429	15.255	36.12	61.95	64.96	73.88	56.12	.2645
SpinQuant+GPTQ	10.867	16.058	33.44	54.34	65.22	75.35	57.06	.3191
FOCUS	9.481	14.035	36.79	65.87	68.62	74.76	59.51	.1803
HiRes	8.889	13.045	38.80	68.56	69.30	76.01	58.64	.1174

Table 4: Complete seven-task results on Qwen3-4B.

Method	WT2↓	C4↓	ARC-C↑	ARC-E↑	HS↑	PIQA↑	Wino↑	Fit-7↓
BF16	13.661	19.824	47.16	77.15	66.73	74.86	57.22	.0000
RTN	18.182	24.181	42.47	69.65	62.12	71.55	56.12	.2283
GPTQ	16.013	22.538	43.48	72.26	62.56	73.23	56.51	.1252
MR-GPTQ	15.528	21.981	46.82	71.89	62.93	73.12	55.96	.0986
BRQ	15.104	21.455	47.16	72.43	63.30	73.94	55.96	.0767
WUSH	14.269	21.132	47.83	73.36	63.37	72.96	57.93	.0479
TORQ	16.998	24.886	41.47	67.42	59.81	71.33	56.27	.1929
BATQuant+GPTQ	17.436	24.937	42.81	68.27	59.83	69.75	55.88	.2051
SpinQuant+GPTQ	14.526	22.073	39.46	63.13	59.63	72.25	55.01	.1409
FOCUS	13.651	21.324	45.82	68.94	63.98	73.12	56.43	.0738
HiRes	14.740	21.117	48.83	74.79	63.71	74.65	57.62	.0524

Table 5: Complete seven-task results on Mistral-7B-v0.3.

Method	WT2↓	C4↓	ARC-C↑	ARC-E↑	HS↑	PIQA↑	Wino↑	Fit-7↓
BF16	5.318	8.439	45.48	73.53	79.60	81.28	70.40	.0000
RTN	6.392	9.898	41.47	68.94	74.94	78.78	64.09	.1512
GPTQ	6.090	9.334	44.48	69.65	75.40	79.43	64.64	.1071
MR-GPTQ	5.731	8.919	45.48	73.11	77.17	80.58	67.88	.0541
BRQ	5.734	8.923	44.48	71.76	77.28	80.20	68.19	.0573
WUSH	5.689	8.834	44.82	72.35	77.20	79.87	68.82	.0503
TORQ	6.120	9.412	39.13	68.18	75.20	79.43	63.93	.1217
BATQuant+GPTQ	6.366	9.775	40.47	69.07	74.13	79.43	64.88	.1482
SpinQuant+GPTQ	6.075	9.299	39.13	67.80	76.14	80.09	64.56	.1142
FOCUS	5.831	9.070	42.14	71.55	77.67	80.20	65.82	.0749
HiRes	5.664	8.810	46.15	73.57	77.56	80.25	67.17	.0452

Table 6: Complete seven-task results on Qwen3-8B.

Method	WT2↓	C4↓	ARC-C↑	ARC-E↑	HS↑	PIQA↑	Wino↑	Fit-7↓
BF16	9.727	15.665	54.85	80.43	73.46	77.69	58.01	.0000
RTN	11.834	18.829	46.82	72.35	67.85	74.48	56.12	.1666
GPTQ	11.554	18.290	47.16	75.04	68.47	74.86	58.41	.1411
MR-GPTQ	10.522	16.831	53.85	79.63	70.22	75.57	58.41	.0587
BRQ	10.701	16.906	54.85	79.50	70.53	76.39	57.38	.0685
WUSH	10.419	16.594	53.85	80.81	71.25	76.22	57.46	.0500
TORQ	12.357	19.117	46.15	74.71	67.31	75.41	56.20	.1970
BATQuant+GPTQ	11.666	18.472	46.82	74.83	68.07	74.81	58.17	.1498
SpinQuant+GPTQ	10.742	17.236	51.84	75.42	67.61	75.79	57.30	.0835
FOCUS	10.423	17.191	46.49	74.03	70.67	75.19	58.33	.1095
HiRes	10.055	16.299	54.85	77.61	70.94	76.33	57.46	.0324

Table 7: Complete seven-task results on Llama-3-8B Base.

Method	WT2↓	C4↓	ARC-C↑	ARC-E↑	HS↑	PIQA↑	Wino↑	Fit-7↓
BF16	6.138	9.597	53.18	76.81	78.50	80.52	62.83	.0000
RTN	8.236	12.650	41.81	69.91	72.37	76.55	59.91	.2522
GPTQ	7.962	12.148	40.47	66.29	73.25	75.63	59.75	.2283
MR-GPTQ	7.249	11.324	45.15	71.21	75.07	77.80	61.40	.1394
BRQ	7.260	11.245	45.15	72.52	75.69	78.94	59.98	.1386
WUSH	7.133	11.117	44.82	73.32	75.38	78.35	61.40	.1248
TORQ	8.253	13.060	44.15	65.53	72.88	75.03	61.09	.2654
BATQuant+GPTQ	8.427	12.999	44.15	67.97	71.16	76.22	59.75	.2727
SpinQuant+GPTQ	8.383	13.125	46.15	71.55	73.27	77.42	62.12	.2588
FOCUS	7.628	11.790	45.82	74.24	75.88	77.37	61.72	.1737
HiRes	7.111	11.026	47.49	72.47	75.24	79.87	60.77	.1187

OLMo-2-13B transfer. We evaluate OLMo-2-1124-13B Base (Walsh et al., 2025) after freezing HiRes, extending the test to a new architecture and model family at 13B parameters. Its results and diagnostics do not enter algorithm design or hyperparameter selection. The calibration and evaluation protocol is unchanged, including disabled KV caching. HiRes achieves the lowest quantized Fit-7 and C4 perplexity, and the highest accuracy on ARC-Challenge, ARC-Easy, HellaSwag, and PIQA (Table 8).

Table 8: Complete seven-task results on OLMo-2-1124-13B Base with the frozen algorithm. Accuracy values are percentages; bold marks the best quantized value in each column. †: topology-safe paired V/O SpinQuant+GPTQ.

Method	WT2↓	C4↓	ARC-C↑	ARC-E↑	HS↑	PIQA↑	Wino↑	Fit-7↓
BF16	5.002954	10.887155	52.84	80.01	82.95	81.77	69.85	.000000
RTN	6.042593	11.842045	48.16	79.00	81.02	80.14	69.22	.135976
GPTQ	6.059833	11.787111	50.50	78.41	80.68	80.09	68.51	.136012
MR-GPTQ	**5.657160**	11.367751	49.50	79.59	81.25	80.36	**70.72**	.084586
BRQ	5.672447	11.388761	52.84	81.19	81.12	81.01	69.69	.081112
WUSH	5.699070	11.375856	54.52	80.85	81.49	80.25	69.85	.082295
TORQ	5.851562	11.587874	52.17	80.30	80.90	80.52	68.19	.106719
BATQuant+GPTQ	6.003139	11.785237	48.16	78.66	81.20	79.98	69.46	.131144
SpinQuant+GPTQ†	6.032943	11.808950	50.50	80.30	80.95	80.79	67.40	.131674
FOCUS	5.911269	11.646977	50.84	78.87	79.63	79.54	67.64	.119531
HiRes	5.658160	**11.338616**	**54.85**	**81.31**	**81.66**	**81.12**	68.19	**.077301**

Qwen3-32B scale transfer. The complete comparison includes BF16 and ten quantized methods under strict OCP MXFP4 W4A4 with E2M1 values, E8M0 shared scales, and contiguous block size 32. KV caching is disabled during evaluation. Calibration uses C4-train, seed 0, and exactly 128×2048 tokens. HiRes achieves the lowest quantized Fit-7 (0.027369), followed by WUSH (0.036489) and BRQ (0.042486). Among quantized methods, it leads on five of seven tasks: WikiText-2/C4 perplexity ($7.926348/13.044736$), ARC-E, HellaSwag, and PIQA. Its Fit-7 is 25.0% lower than WUSH's (Table 9).

Table 9: Complete Qwen3-32B seven-task results under strict MXFP4 W4A4 with KV caching disabled. Accuracy values are fractions. Quantized methods are ordered by Fit-7; bold marks the best quantized value in each column.

Method	WT2↓	C4↓	ARC-C↑	ARC-E↑	HS↑	PIQA↑	Wino↑	Fit-7↓
BF16	7.610919	12.689191	0.595318	0.826178	0.815674	0.819913	0.614049	0.000000
HiRes	**7.926348**	**13.044736**	0.575251	**0.832071**	**0.810396**	**0.815016**	0.623520	**0.027369**
WUSH	7.950299	13.118806	0.565217	0.816919	0.804123	0.804135	0.624309	0.036489
BRQ	8.138484	13.301107	**0.605351**	0.819865	0.801932	0.807399	**0.632991**	0.042486
MR-GPTQ	8.195791	13.484847	0.581940	0.830808	0.799243	0.786181	0.627466	0.052413
FOCUS	8.141298	13.909520	0.581940	0.815236	0.789484	0.792709	0.614049	0.067147
GPTQ	8.420178	13.762802	0.578595	0.801768	0.793467	0.804679	0.610103	0.074650
RTN	8.614437	14.146650	0.568562	0.813131	0.793169	0.796518	0.614049	0.091896
TORQ	8.667482	14.188477	0.595318	0.805135	0.797849	0.790533	0.621152	0.092880
SpinQuant+GPTQ	8.416143	13.844641	0.515050	0.783670	0.774547	0.791077	0.602210	0.102274
BATQuant+GPTQ	8.973184	14.626795	0.575251	0.796296	0.789385	0.792709	0.624309	0.121656

C.2 Calibration and Evaluation Sensitivity

Calibration seeds. We repeat calibration with seeds $\{0, 3, 5\}$ on Qwen3-8B and Meta-Llama-3-8B, abbreviated as Qwen and Llama in the tables below. Seed 0 matches the main comparison. Each configuration collects its own recovery statistics and reruns downstream evaluation. Table captions distinguish Fit-3 (WT2, C4, ARC-C; Eq. 16) from Fit-7 (Eq. 12); both use the BF16 references in Table 10.

Before running the comparisons, we specified minimum differences of 0.002 for Fit-7 and 0.003 for Fit-3. These thresholds apply to each hypothesis's designated comparison between the full method and its strongest control. Intervals use the $n = 3$ t distribution, with $t_{0.975,2} = 4.303$; paired directions and intervals are reported together. Accuracy sample counts are 299 for ARC-C, 2376 for ARC-E, 10042 for HellaSwag, 1838 for PIQA, and 1267 for WinoGrande.

Table 10: BF16 references for the multi-seed ablations. Accuracy is reported as a fraction.

Model	WT2↓	C4↓	ARC-C↑	ARC-E↑	Hella↑	PIQA↑	Wino↑
Qwen	9.726731	15.665062	0.548495	0.804293	0.734615	0.776931	0.580110
Llama	6.138148	9.597075	0.531773	0.768098	0.785003	0.805223	0.628256

HiRes improves on WUSH in $3/3$ seeds on each model (Table 11). On Qwen3-8B, mean$\pm$SD Fit-7 is 0.032692 ± 0.000298 for HiRes and 0.049776 ± 0.002902 for WUSH. The mean paired difference is -0.017085, with SD 0.002763 and 95% CI $[-0.023948, -0.010221]$. On Llama-3-8B, the corresponding means are 0.116158 ± 0.004006 and 0.121639 ± 0.003560. The mean paired difference is -0.005481, with SD 0.001108 and 95% CI $[-0.008233, -0.002729]$. Table 17 also lists the per-seed paired differences.

Table 11: Calibration-seed stability under the seven-task protocol. Accuracy columns are fractions; lower Fit-7 is better.

Model	Method	Seed	WT2↓	C4↓	ARC-C↑	ARC-E↑
Qwen	HiRes	0	10.055217	16.298661	0.548495	0.776094
Qwen	HiRes	3	10.159826	16.315420	0.535117	0.796296
Qwen	HiRes	5	10.149540	16.306193	0.548495	0.774411
Qwen	WUSH	0	10.418540	16.594240	0.538462	0.808081
Qwen	WUSH	3	10.378339	16.552246	0.541806	0.789562
Qwen	WUSH	5	10.446173	16.615181	0.541806	0.802609
Llama	HiRes	0	7.110953	11.026218	0.474916	0.724747
Llama	HiRes	3	7.115418	10.994374	0.464883	0.751684
Llama	HiRes	5	7.101174	10.987771	0.498328	0.755471
Llama	WUSH	0	7.132691	11.116994	0.448161	0.733165
Llama	WUSH	3	7.137809	11.097389	0.461538	0.737795
Llama	WUSH	5	7.130909	11.071733	0.474916	0.760943

Model	Method	Seed	HellaSwag↑	PIQA↑	WinoGrande↑	Fit-7↓
Qwen	HiRes	0	0.709420	0.763330	0.574586	0.032409
Qwen	HiRes	3	0.711512	0.779652	0.581689	0.032663
Qwen	HiRes	5	0.712906	0.762242	0.587214	0.033003
Qwen	WUSH	0	0.712507	0.762242	0.574586	0.050030
Qwen	WUSH	3	0.710914	0.760609	0.592739	0.046756
Qwen	WUSH	5	0.704342	0.762786	0.574586	0.052543
Llama	HiRes	0	0.752440	0.798694	0.607735	0.118743
Llama	HiRes	3	0.750747	0.791621	0.610892	0.118187
Llama	HiRes	5	0.754431	0.798694	0.606156	0.111543
Llama	WUSH	0	0.753834	0.783460	0.614049	0.124760
Llama	WUSH	3	0.751344	0.794342	0.611681	0.122394
Llama	WUSH	5	0.753933	0.789989	0.611681	0.117762

Alternative summaries. We also summarize the seven-task endpoints by their mean relative damage and the equally weighted mean of the five accuracy tasks:

$$\mathrm{MeanDamage}_m = \frac{1}{7}\sum_{j=1}^{7} d_{mj}, \qquad \mathrm{Acc5}_m = \frac{100}{5}\sum_{j\in\mathcal{A}} \mathrm{Acc}_{mj}, \tag{39}$$

where $\mathcal{A}$ contains ARC-C, ARC-E, HellaSwag, PIQA, and WinoGrande, and accuracies inside the sum are fractions. Damage follows Eq. 12, including negative values for gains over BF16. Fit-7 retains the reported table values; the alternative aggregates are recomputed from the same seven-task endpoints.

HiRes lowers MeanDamage on all seven models (Table 12). On Qwen3-8B, WT2, C4, and Fit-7 improve while Acc5 is slightly lower. The completed Qwen3-32B evaluation gives MeanDamage 0.022416 for WUSH and 0.013294 for HiRes. Their worst-task damages are 0.05056 on ARC-C and 0.04144 on WT2, respectively; the seven endpoints appear in Table 9.

Table 12: Alternative aggregates from the frozen seven-task evaluations at calibration seed 0. All results use the complete 299-question ARC-C validation split.

Model	Method	Fit-7↓	MeanDamage↓	Acc5 (%)↑
Llama-3.2-3B	WUSH	0.122400	0.080676	61.5880
	HiRes	0.117400	0.071644	62.2620
Qwen3-4B	WUSH	0.047900	0.029819	63.0900
	HiRes	0.052400	0.025780	63.9200
Mistral-7B	WUSH	0.050300	0.031010	68.6120
	HiRes	0.045200	0.025419	68.9400
Qwen3-8B	WUSH	0.050030	0.028935	67.9176
	HiRes	0.032409	0.024372	67.4385
Llama-3-8B	WUSH	0.124760	0.087494	66.6534
	HiRes	0.118743	0.079002	67.1706
OLMo-2-13B	WUSH	0.082295	0.025448	73.3907
	HiRes	0.077301	0.023638	73.4266
Qwen3-32B	WUSH	0.036489	0.022416	72.2941
	HiRes	0.027369	0.013294	73.1251

C.3 Stage and Geometry Ablations

We first add the three stages in deployment order, then isolate the geometric corrections at fixed downstream procedures. The stage comparison in Table 13 reports Fit-3 on Qwen3-8B and Meta-Llama-3-8B. Each stage lowers damage on both models.

Table 13: Post-discovery stage ablations with raw endpoints. Each row retains the preceding stages. The Discrete row corresponds to the configuration with both structural recovery paths disabled.

Model	Program	WT2↓	C4↓	ARC-C↑	Mean dmg.↓	Worst dmg.↓	Fit-3↓
Qwen3-8B	Strict RTN	11.8342	18.8285	46.82	.18832	.21667	.20249
	+ Geometric shaping	10.5374	16.7494	51.17	.07321	.08335	.07828
	+ Discrete realization	10.1858	16.4524	50.84	.05688	.07317	.06502
	+ Structural recovery	10.0552	16.2987	54.85	.02474	.04045	.03259
Llama-3-8B	Strict RTN	8.2362	12.6497	41.81	.29124	.34181	.31653
	+ Geometric shaping	7.1963	11.2766	46.15	.15982	.17500	.16741
	+ Discrete realization	7.1417	11.1163	46.82	.14710	.16349	.15530
	+ Structural recovery	7.1110	11.0262	47.49	.13811	.15849	.14830

Table 14: Geometric-component ablations at calibration seed 0. Downstream reconstruction and recovery procedures are fixed.

Model	Configuration	WT2↓	C4↓	ARC-C↑	Fit-3↓
Qwen3-8B	Full HiRes	10.0552	16.2987	54.85	.03259
	Remove local interaction K	10.1215	16.3613	53.51	.04046
	$D = I$	10.1480	16.3925	53.18	.04326
	$K = 0, D = I$	10.2436	16.5013	52.17	.05258
Llama-3-8B	Full HiRes	7.1110	11.0262	47.49	.14830
	Remove local interaction K	7.1393	11.0706	46.49	.15529
	$D = I$	7.1525	11.0931	45.82	.15921
	$K = 0, D = I$	7.1813	11.1419	44.48	.16738

The local interaction K improves fitness on both models, by 0.00787 on Qwen3-8B and 0.00699 on Llama-3-8B. Diagonal balance contributes on both as well: removing D degrades Qwen3-8B by 0.01067 and Llama-3-8B by 0.01091, and removing both is worse than removing either alone.

Removing either geometric correction increases damage (Table 15). In all $6/6$ model–seed pairs, the full method is best, followed by removing K, removing D, and removing both. Relative to removing K, the strongest control, the full method lowers Fit-7 by 0.005259–0.006581 on Qwen and 0.005361–0.006121 on Llama. Both ranges exceed the 0.002 threshold. Removing both corrections is worse than removing either alone. Each correction also lowers Fit-7 when used without the other.

Table 15: Geometry ablations across calibration seeds. Reference denotes $K = 0$, $D = I$. Each configuration refits recovery statistics on its own encoded state. Accuracy is reported as a fraction.

Model	Config.	Seed	WT2↓	C4↓	ARC-C↑	ARC-E↑
Qwen	Full	0	10.055217	16.298661	0.548495	0.776094
Qwen	$K = 0$	0	10.121483	16.361274	0.535117	0.773569
Qwen	$D = I$	0	10.148022	16.392512	0.531773	0.771886
Qwen	Reference	0	10.243630	16.501342	0.521739	0.766414
Qwen	Full	3	10.159826	16.315420	0.535117	0.796296
Qwen	$K = 0$	3	10.224922	16.376356	0.521739	0.793771
Qwen	$D = I$	3	10.244220	16.399973	0.518395	0.792508
Qwen	Reference	3	10.314493	16.480255	0.515050	0.788300
Qwen	Full	5	10.149540	16.306193	0.548495	0.774411
Qwen	$K = 0$	5	10.208067	16.361005	0.535117	0.772306
Qwen	$D = I$	5	10.230606	16.387448	0.535117	0.770623
Qwen	Reference	5	10.332922	16.501719	0.521739	0.765152
Llama	Full	0	7.110953	11.026218	0.474916	0.724747
Llama	$K = 0$	0	7.139273	11.070583	0.464883	0.719697
Llama	$D = I$	0	7.152459	11.093144	0.458194	0.715909
Llama	Reference	0	7.181305	11.141896	0.444816	0.708754
Llama	Full	3	7.115418	10.994374	0.464883	0.751684
Llama	$K = 0$	3	7.147299	11.044141	0.454849	0.745791
Llama	$D = I$	3	7.155975	11.059539	0.448161	0.743266
Llama	Reference	3	7.174092	11.090511	0.441472	0.738215
Llama	Full	5	7.101174	10.987771	0.498328	0.755471
Llama	$K = 0$	5	7.127631	11.029130	0.488294	0.750842
Llama	$D = I$	5	7.140326	11.050767	0.481605	0.747054
Llama	Reference	5	7.169855	11.100461	0.468227	0.739899

Table 15: Geometry ablations across calibration seeds (continued).

Model	Config.	Seed	HellaSwag↑	PIQA↑	WinoGrande↑	Fit-7↓
Qwen	Full	0	0.709420	0.763330	0.574586	0.032409
Qwen	$K = 0$	0	0.707628	0.760609	0.573796	0.037668
Qwen	$D = I$	0	0.706831	0.758977	0.573007	0.039912
Qwen	Reference	0	0.704441	0.755713	0.569061	0.047395
Qwen	Full	3	0.711512	0.779652	0.581689	0.032663
Qwen	$K = 0$	3	0.709819	0.776931	0.580900	0.039244
Qwen	$D = I$	3	0.709221	0.775843	0.580110	0.042126
Qwen	Reference	3	0.707429	0.773667	0.576953	0.047629
Qwen	Full	5	0.712906	0.762242	0.587214	0.033003
Qwen	$K = 0$	5	0.711312	0.760065	0.586425	0.039073
Qwen	$D = I$	5	0.710715	0.758433	0.585635	0.040972
Qwen	Reference	5	0.708126	0.754625	0.581689	0.050820
Llama	Full	0	0.752440	0.798694	0.607735	0.118743
Llama	$K = 0$	0	0.750548	0.795430	0.604578	0.124348
Llama	$D = I$	0	0.749054	0.792165	0.602210	0.127689
Llama	Reference	0	0.746166	0.787813	0.598264	0.134297
Llama	Full	3	0.750747	0.791621	0.610892	0.118187
Llama	$K = 0$	3	0.748656	0.787813	0.607735	0.124308
Llama	$D = I$	3	0.747461	0.785092	0.605367	0.126983
Llama	Reference	3	0.745569	0.782372	0.602999	0.131471
Llama	Full	5	0.754431	0.798694	0.606156	0.111543
Llama	$K = 0$	5	0.752639	0.795430	0.602999	0.116904
Llama	$D = I$	5	0.751245	0.792709	0.600631	0.120135
Llama	Reference	5	0.748257	0.787813	0.596685	0.126871

Geometry replacement. To separate geometry from downstream corrections, we compare WUSH, WUSH+R, and the complete HiRes on Qwen3-8B and Meta-Llama-3-8B, using calibration seeds $0, 3, 5$. WUSH+R uses WUSH geometry through a common transform interface, rebuilds W4 in those coordinates, and applies the same one-code refinement and attention-to-MLP recovery as HiRes. Both configurations use GPTQ damping 0.01, FP32 reconstruction, and the same 128×2048 C4 calibration sequences and order. They share E2M1/E8M0 Block32 encoding, the one-code rule, and solver budgets.

Each geometry is followed by freshly fitted attention and then MLP recovery, with two affine coefficients per KV head and three MLP coefficients per layer; σ_g is a separate scale statistic. The comparison fixes refinement and recovery procedures while changing the geometry and the encoded state it produces.

With refinement and recovery disabled, the common interface reproduces WUSH in all six model–seed configurations: W4 codes and E8M0 exponents match elementwise, the five accuracy tasks give identical correctness vectors, and WT2, C4, and Fit-7 agree at reporting precision. GPTQ ordering, calibration files, and data order are unchanged. We reuse these WUSH and HiRes results and add WUSH+R.

Adding refinement and recovery to WUSH lowers mean Fit-7 by 0.010784 on Qwen and 0.002878 on Llama. With the same recovery capacity, HiRes lowers it by another 0.006301 and 0.002603, respectively (Table 16). Both steps improve Fit-7 in all three paired seeds on each model (Table 17). Mean WT2 and C4 perplexity decrease at both steps on both models. Mean accuracy across the five classification tasks decreases slightly on Qwen and increases on Llama.

Table 16: Geometry replacement with matched refinement and recovery. WT2, C4, and Acc5 are arithmetic means over calibration seeds $0, 3, 5$. Acc5 averages the five accuracy tasks equally within each seed before averaging across seeds. Fit-7 is computed per seed; $\pm$ denotes sample SD.

Model	Method	Fit-7↓	WT2↓	C4↓	Acc5 (%)↑
Qwen	WUSH	0.049776 ± 0.002902	10.414351	16.587222	67.8509
	WUSH+R	0.038992 ± 0.002229	10.239410	16.428239	67.7899
	HiRes	0.032692 ± 0.000298	10.121528	16.306758	67.7431
Llama	WUSH	0.121639 ± 0.003560	7.133803	11.095372	67.2055
	WUSH+R	0.118761 ± 0.003275	7.121359	11.044775	67.4389
	HiRes	0.116158 ± 0.004006	7.109182	11.002788	67.6096

Table 17: Paired Fit-7 differences for geometry replacement. Negative values favor the first method. Intervals use the reported six-decimal differences, $n = 3$, and $t_{0.975,2} = 4.303$.

Model	Difference	Seed 0↓	Seed 3↓	Seed 5↓	Mean [95% CI]↓
Qwen	WUSH+R − WUSH	−0.012458	−0.008913	−0.010981	−0.010784 [−0.015207, −0.006361]
	HiRes − WUSH+R	−0.005163	−0.005180	−0.008559	−0.006301 [−0.011159, −0.001443]
	HiRes − WUSH	−0.017621	−0.014093	−0.019540	−0.017085 [−0.023948, −0.010221]
Llama	WUSH+R − WUSH	−0.003866	−0.001996	−0.002772	−0.002878 [−0.005212, −0.000544]
	HiRes − WUSH+R	−0.002151	−0.002211	−0.003447	−0.002603 [−0.004421, −0.000786]
	HiRes − WUSH	−0.006017	−0.004207	−0.006219	−0.005481 [−0.008233, −0.002729]

C.4 Activation-Code Refinement

Sequential reconstruction and one-code refinement both improve Fit-3 (Table 18), with the larger reduction coming from sequential reconstruction. We next isolate the activation-code rule while retaining the same weight reconstruction.

Table 18: Discrete-realization ablations at calibration seed 0. Geometry and structural recovery procedures are fixed.

Model	Configuration	WT2↓	C4↓	ARC-C↑	Fit-3↓
Qwen3-8B	Strict RTN realization	10.2749	16.4117	49.83	.07831
	Sequential reconstruction	10.1817	16.3898	53.18	.04397
	Full discrete realization	10.0552	16.2987	54.85	.03259
Llama-3-8B	Strict RTN realization	7.2654	11.3159	43.48	.18269
	Sequential reconstruction	7.1464	11.0486	47.16	.15359
	Full discrete realization	7.1110	11.0262	47.49	.14830

We compare nearest rounding, sensitivity-only refinement, and the full mismatch-aware rule from Appendix B.3. Payload format, geometry, weight reconstruction, and structural recovery are fixed. Nearest rounding corresponds to the sequential-reconstruction row in Table 18; the full rule gives HiRes.

The full rule has lower Fit-7 than both controls in all $6/6$ model–seed pairs (Table 19). Its mean gain over nearest rounding and sensitivity-only is, respectively, 0.006632 and 0.010177 on Qwen, and 0.005482 and 0.007635 on Llama. The gains range over 0.0055–0.0109 on Qwen and 0.0050–0.0081 on Llama. All endpoint directions also favor the full rule across the 12 configuration comparisons, covering 84 task-level differences across the paired seeds.

Table 19: One-code rule ablations with the same payload format and remaining pipeline. Nearest uses nearest rounding; Sensitivity omits the realized-weight mismatch term; Full includes it.

Model	Method	Seed	WT2↓	C4↓	ARC-C↑	ARC-E↑
Qwen	Nearest	0	10.181742	16.389816	0.531773	0.774411
Qwen	Sensitivity	0	10.234345	16.416364	0.541806	0.772727
Qwen	Full	0	10.055217	16.298661	0.548495	0.776094
Qwen	Nearest	3	10.234382	16.368622	0.525084	0.795034
Qwen	Sensitivity	3	10.308024	16.411895	0.528428	0.793771
Qwen	Full	3	10.159826	16.315420	0.535117	0.796296
Qwen	Nearest	5	10.213381	16.351769	0.541806	0.773148
Qwen	Sensitivity	5	10.282356	16.392694	0.541806	0.771886
Qwen	Full	5	10.149540	16.306193	0.548495	0.774411
Llama	Nearest	0	7.146382	11.048571	0.471572	0.720960
Llama	Sensitivity	0	7.153818	11.067518	0.458194	0.719276
Llama	Full	0	7.110953	11.026218	0.474916	0.724747
Llama	Nearest	3	7.146486	11.013903	0.451505	0.747475
Llama	Sensitivity	3	7.153895	11.031315	0.451505	0.746633
Llama	Full	3	7.115418	10.994374	0.464883	0.751684
Llama	Nearest	5	7.131137	11.006632	0.488294	0.751263
Llama	Sensitivity	5	7.142052	11.027072	0.481605	0.750000
Llama	Full	5	7.101174	10.987771	0.498328	0.755471

Table 19: Activation-code refinement across calibration seeds (continued).

Model	Method	Seed	HellaSwag↑	PIQA↑	WinoGrande↑	Fit-7↓
Qwen	Nearest	0	0.708425	0.758433	0.572218	0.040086
Qwen	Sensitivity	0	0.708524	0.759521	0.571429	0.042128
Qwen	Full	0	0.709420	0.763330	0.574586	0.032409
Qwen	Nearest	3	0.710715	0.775299	0.580110	0.039377
Qwen	Sensitivity	3	0.710715	0.776387	0.579321	0.043574
Qwen	Full	3	0.711512	0.779652	0.581689	0.032663
Qwen	Nearest	5	0.712209	0.758977	0.585635	0.038507
Qwen	Sensitivity	5	0.712209	0.759521	0.584846	0.042905
Qwen	Full	5	0.712906	0.762242	0.587214	0.033003
Llama	Nearest	0	0.751842	0.794342	0.605367	0.123719
Llama	Sensitivity	0	0.751643	0.793254	0.603788	0.126800
Llama	Full	0	0.752440	0.798694	0.607735	0.118743
Llama	Nearest	3	0.750050	0.786725	0.607735	0.124270
Llama	Sensitivity	3	0.749950	0.786725	0.607735	0.125177
Llama	Full	3	0.750747	0.791621	0.610892	0.118187
Llama	Nearest	5	0.753734	0.794342	0.603788	0.116931
Llama	Sensitivity	5	0.753635	0.793254	0.602210	0.119400
Llama	Full	5	0.754431	0.798694	0.606156	0.111543

The local objectives explain why sensitivity alone is insufficient in these comparisons (Table 20). Sensitivity-only achieves the lowest value of its own objective, $0.912/0.921$ on Qwen/Llama, but raises the mismatch term to $1.043/1.036$. The full rule accepts sensitivity terms of $0.947/0.953$ while lowering mismatch to $0.918/0.926$, and obtains the best endpoints. Local errors are measured on all layers using refinement. Each layer's error is normalized by its output-element count, then averaged across layers. Measurement segments are independent of, and disjoint from, the rule-fitting data.

Table 20: Local rule objectives at seed 0, each normalized to nearest rounding. Lower is better for both objectives and Fit-7.

Model	Rule	Sensitivity-weighted local error↓	Mismatch-weighted local error↓	Fit-7↓
Qwen	Nearest	1.000	1.000	0.040086
Qwen	Sensitivity	0.912	1.043	0.042128
Qwen	Full	0.947	0.918	0.032409
Llama	Nearest	1.000	1.000	0.123719
Llama	Sensitivity	0.921	1.036	0.126800
Llama	Full	0.953	0.926	0.118743

The observed edit rate is below the $1/32 = 3.125\%$ element-level ceiling (Table 21). Qwen changes an average of 0.61 codes per Block32, affecting 1.91% of A4 elements; Llama changes 0.68, affecting 2.12%. Table 22 separates one-time quantization cost from recurring decoding overhead, measured against the same W4A4 model with one-code refinement disabled.

Table 21: Observed one-code edits. Codes per block and the fraction of all A4 elements use different denominators.

Model	Edited codes per Block32	Edited A4 elements	Rule ceiling
Qwen3-8B	0.61	1.91%	3.125%
Llama-3-8B	0.68	2.12%	3.125%

Table 22: One-time quantization cost (upper panel) and per-forward decoding overhead (lower panel) of one-code refinement. Decoding uses batch=1, seq=2048, and greedy generation. Measurements use a single NVIDIA A100-SXM4-80GB GPU with the PyTorch 2.9.1 (CUDA 12.6) eager/reference fake-quant backend.

Model	Without refinement↓	With refinement↓	Change↓
Qwen3-8B	58.2 min	61.6 min	+3.4 min (+5.8%)
Llama-3-8B	54.7 min	57.8 min	+3.1 min (+5.7%)

Model	Without refinement↑	With refinement↑	Change↑
Qwen3-8B	41.7 tok/s	40.6 tok/s	−2.6%
Llama-3-8B	43.2 tok/s	42.1 tok/s	−2.5%

C.5 Recovery Paths and Remeasurement

We test which paths benefit from recovery and when their fitting statistics should be collected. The path ablations keep per-path capacity fixed; the timing controls keep both operators fixed and change their fitting state.

Table 23: Structural-recovery factorization and fresh-statistics control.

Model	Configuration	WT2↓	C4↓	ARC-C↑	Fit-3↓
Qwen3-8B	Both paths disabled	10.1858	16.4524	50.84	.06502
	Attention only	10.1588	16.4411	52.17	.04856
	MLP only	10.0965	16.3397	53.51	.03911
	Both paths	10.0552	16.2987	54.85	.03259
	Stale MLP statistics	10.1090	16.3722	53.18	.04173
Llama-3-8B	Both paths disabled	7.1417	11.1163	46.82	.15530
	Attention only	7.1328	11.0855	46.49	.15484
	MLP only	7.1263	11.0618	46.49	.15373
	Both paths	7.1110	11.0262	47.49	.14830
	Stale MLP statistics	7.1454	11.0816	45.82	.15824

Attention and MLP paths. The full pair is best on both models, improving over the stronger single path by 0.00652 on Qwen3-8B and 0.00543 on Llama-3-8B. On Qwen3-8B each single path also improves clearly over disabling both. On Llama-3-8B the two single paths lie within 0.0016 of the no-recovery control and within 0.0012 of each other. The full pair gives the largest separation from the no-recovery control. Each comparison varies path presence at fixed per-path capacity. The stale control measures the MLP recovery statistics on the Stage-II parent before installing the attention correction, while retaining the normal attention-to-MLP deployment order.

Both recovery paths give the lowest damage across seeds (Table 24). In all $6/6$ model–seed pairs, damage increases in the order: both paths, MLP only, attention only, and neither path. The full pair improves over MLP-only recovery, the strongest single-path control, by 0.003562–0.004311 on Qwen and 0.002875–0.003193 on Llama. The two single paths differ by only 0.000289–0.000646 on Llama, versus 0.002494–0.006380 on Qwen. Their Llama gains over no recovery are 0.001628–0.002464. The full pair has the largest gain, with per-path capacity fixed across the compared configurations.

Table 24: Recovery-path ablations across calibration seeds. Per-path capacity is fixed, and each configuration refits recovery statistics. Accuracy is reported as a fraction.

Model	Config.	Seed	WT2↓	C4↓	ARC-C↑	ARC-E↑
Qwen	None	0	10.185825	16.452397	0.508361	0.783670
Qwen	Attention	0	10.158764	16.441078	0.521739	0.783670
Qwen	MLP	0	10.096486	16.339713	0.535117	0.774411
Qwen	Both	0	10.055217	16.298661	0.548495	0.776094
Qwen	None	3	10.264299	16.437247	0.505017	0.802189
Qwen	Attention	3	10.231741	16.413413	0.515050	0.801347
Qwen	MLP	3	10.196311	16.351377	0.521739	0.794613
Qwen	Both	3	10.159826	16.315420	0.535117	0.796296
Qwen	None	5	10.286865	16.466401	0.505017	0.782407
Qwen	Attention	5	10.231431	16.417828	0.528428	0.780303
Qwen	MLP	5	10.187542	16.343661	0.535117	0.772727
Qwen	Both	5	10.149540	16.306193	0.548495	0.774411
Llama	None	0	7.141691	11.116290	0.468227	0.729798
Llama	Attention	0	7.132786	11.085461	0.464883	0.734428
Llama	MLP	0	7.126336	11.061782	0.464883	0.727273
Llama	Both	0	7.110953	11.026218	0.474916	0.724747
Llama	None	3	7.143750	11.077097	0.458194	0.756313
Llama	Attention	3	7.136955	11.052609	0.454849	0.761364
Llama	MLP	3	7.133187	11.035307	0.454849	0.754630
Llama	Both	3	7.115418	10.994374	0.464883	0.751684
Llama	None	5	7.133531	11.082381	0.491639	0.760943
Llama	Attention	5	7.127492	11.059036	0.488294	0.767256
Llama	MLP	5	7.119638	11.030367	0.488294	0.758418
Llama	Both	5	7.101174	10.987771	0.498328	0.755471

Table 24: Recovery-path ablations across calibration seeds (continued).

Model	Config.	Seed	HellaSwag↑	PIQA↑	WinoGrande↑	Fit-7↓
Qwen	None	0	0.707927	0.760065	0.576953	0.055139
Qwen	Attention	0	0.709221	0.764962	0.581689	0.040171
Qwen	MLP	0	0.708823	0.761153	0.577743	0.035971
Qwen	Both	0	0.709420	0.763330	0.574586	0.032409
Qwen	None	3	0.710317	0.776931	0.583268	0.054925
Qwen	Attention	3	0.711412	0.780740	0.586425	0.043354
Qwen	MLP	3	0.711014	0.777476	0.584846	0.036974
Qwen	Both	3	0.711512	0.779652	0.581689	0.032663
Qwen	None	5	0.711312	0.758977	0.589582	0.057757
Qwen	Attention	5	0.712707	0.763330	0.592739	0.039652
Qwen	MLP	5	0.712408	0.760065	0.590371	0.037158
Qwen	Both	5	0.712906	0.762242	0.587214	0.033003
Llama	None	0	0.748357	0.788357	0.606946	0.124082
Llama	Attention	0	0.752141	0.793254	0.607735	0.122174
Llama	MLP	0	0.751344	0.797062	0.609313	0.121618
Llama	Both	0	0.752440	0.798694	0.607735	0.118743
Llama	None	3	0.746963	0.781828	0.610103	0.123211
Llama	Attention	3	0.750448	0.786181	0.610892	0.121583
Llama	MLP	3	0.749552	0.789989	0.612470	0.121294
Llama	Both	3	0.750747	0.791621	0.610892	0.118187
Llama	None	5	0.750149	0.787813	0.605367	0.117094
Llama	Attention	5	0.754133	0.792165	0.606156	0.115382
Llama	MLP	5	0.753137	0.797062	0.607735	0.114736
Llama	Both	5	0.754431	0.798694	0.606156	0.111543

Remeasurement after attention. With W4/A4 operators already installed in both conditions, we vary whether MLP statistics are measured before or after the attention correction. Remeasuring after attention improves WT2, C4, and ARC-C individually in all $6/6$ model–seed pairs (Table 25). The Fit-3 reduction is 0.006444–0.009132 on Qwen and 0.009304–0.012954 on Llama. For Qwen, fresh-minus-stale differences have mean -0.008022, SD 0.001404, and 95% CI $[-0.011509, -0.004535]$. For Llama, the mean is -0.010734, SD 0.001949, and 95% CI $[-0.015576, -0.005891]$. Each model has $3/3$ negative differences. The next control instead varies whether recovery is fitted before or after A4 encoding and measures both local reconstruction error and task endpoints.

Table 25: MLP statistics measured after (fresh) or before (stale) attention recovery. The last column is paired fresh-minus-stale Fit-3; both conditions retain attention-to-MLP deployment.

Model	Seed	Statistics	WT2↓	C4↓	ARC-C↑	Fit-3↓	Δ Fit-3↓
Qwen	0	Fresh	10.055217	16.298661	0.548495	0.032593	-0.009132
Qwen	0	Stale	10.108964	16.372215	0.531773	0.041725	–
Qwen	3	Fresh	10.159826	16.315420	0.535117	0.040669	-0.006444
Qwen	3	Stale	10.211284	16.398624	0.528428	0.047113	–
Qwen	5	Fresh	10.149540	16.306193	0.548495	0.035800	-0.008489
Qwen	5	Stale	10.203715	16.372940	0.535117	0.044289	–
Llama	0	Fresh	7.110953	11.026218	0.474916	0.148296	-0.009943
Llama	0	Stale	7.145382	11.081647	0.458194	0.158239	–
Llama	3	Fresh	7.115418	10.994374	0.464883	0.151372	-0.009304
Llama	3	Stale	7.148216	11.083544	0.451505	0.160676	–
Llama	5	Fresh	7.101174	10.987771	0.498328	0.139228	-0.012954
Llama	5	Stale	7.139862	11.069215	0.474916	0.152182	–

Remeasurement after encoding. The recovery operators and parameter counts are identical in both arms: attention uses two affine coefficients $(a_{\ell h}, b_{\ell h})$ per key–value head (Eq. 9), and MLP uses three coefficients (α, β, η) per layer (Eq. 10). Both use the same weighted ridge fit, median-deviation shrinkage, and projection to $\|[\alpha, \beta, \eta]\|_1 \leq 1/8$ for MLP. We change only whether the fitting second moments are collected before A4 encoding or after actual MXFP4 encoding.

Post-encoding statistics improve both the deployed local error and the task endpoints (Table 26). Both arms measure MSE in the same actual encoded state, relative to BF16 layer outputs. We include every layer with attention V/O or MLP recovery, normalize each layer's squared error by its output-element count, and take the arithmetic mean across layers. The measurement set contains 512 independent sequences of length 2048, disjoint from coefficient-fitting data. The before column measures the common model without recovery; the after column applies the coefficients fitted by each arm.

Table 26: Recovery fitted before or after A4 encoding at seed 0: task endpoints (upper panel) and MSE in the actual encoded state (lower panel). The operator forms and coefficient counts are fixed.

Model	Statistics	WT2↓	C4↓	ARC-C↑	Fit-3↓
Qwen	pre-encoding	10.142501	16.451416	0.521739	0.048720
Qwen	post-encoding	10.055217	16.298661	0.548495	0.032593
Llama	pre-encoding	7.148913	11.122943	0.464883	0.157243
Llama	post-encoding	7.110953	11.026218	0.474916	0.148296

Model	Statistics	Local MSE before↓	Local MSE after↓	Reduction↑
Qwen	pre-encoding	0.0030302292	0.0030296451	0.019%
Qwen	post-encoding	0.0030302292	0.0027418735	9.52%
Llama	pre-encoding	4.1640092e-05	4.1635055e-05	0.012%
Llama	post-encoding	4.1640092e-05	3.8104617e-05	8.49%

Coefficients fitted before encoding barely change deployed MSE: the reductions are 0.019% on Qwen and 0.012% on Llama. Fitting after encoding gives 9.52% and 8.49% reductions with the same operator forms, counts, and locations. Across calibration seeds, post-encoding fitting lowers

Fit-3 by 0.016127–0.016479 on Qwen and 0.008312–0.009614 on Llama, above the 0.003 threshold (Table 27).

Table 27: Encoding-boundary control across seeds. WT2, C4, and ARC-C are pre-encoding-fit endpoints; post-encoding-fit endpoints appear in Table 25 (fresh rows).

Model	Seed	Pre WT2↓	Pre C4↓	Pre ARC-C↑	Pre Fit-3↓	Post Fit-3↓	Gain↑
Qwen	0	10.142501	16.451416	0.521739	0.048720	0.032593	0.016127
Qwen	3	10.220455	16.420544	0.515050	0.057148	0.040669	0.016479
Qwen	5	10.243144	16.468566	0.521739	0.052074	0.035800	0.016274
Llama	0	7.148913	11.122943	0.464883	0.157243	0.148296	0.008947
Llama	3	7.157778	11.101946	0.454849	0.160986	0.151372	0.009614
Llama	5	7.134994	11.073777	0.488294	0.147540	0.139228	0.008312

C.6 Computational and Storage Costs

We compare complete HiRes with BF16 for persistent storage and full-forward latency, and report one-time quantization separately. The storage and latency results use the optimized inference layout and fused online operations below.

Persistent storage. Each target linear layer retains its own Block32 matrices: the folded input transform $U = DT$, the upper triangle of G_Q including its diagonal, and the full K_Q, all in FP32. Symmetric entries of G_Q are recovered by mirroring the stored values. Each input block therefore uses $4(1024 + 528 + 1024) = 10{,}304$ bytes of matrix state. Attention recovery adds two coefficients per key–value head; MLP recovery adds $\alpha, \beta, \eta, \sigma_g$ per layer. Their FP32 storage totals 2,880, 2,560, and 5,120 bytes for Qwen3-8B, Llama-3-8B, and Qwen3-32B, respectively. Construction-only inverses, intermediate geometry, calibration arrays, Hessians, and BF16 target-weight copies are released after quantization. Table 28 counts decoded BF16 parameters and persistent auxiliary tensors; activations, temporary workspaces, and allocator reserves are outside this static accounting. The total payload exceeds the BF16 parameter payload by 1.87%–2.61%.

Table 28: Static storage in the BF16-decoded implementation. Auxiliary storage includes all retained matrices and recovery coefficients. Blocks are counted separately for each target linear layer; MiB and GiB denote 2^{20} and 2^{30} bytes.

Model	Input blocks	BF16 (GiB)	Auxiliary (MiB)	HiRes total (GiB)	Increase
Qwen3-8B	41,472	15.256	407.534	15.654	2.61%
Llama-3-8B	38,912	14.958	382.377	15.331	2.50%
Qwen3-32B	118,784	61.024	1167.255	62.164	1.87%

Full-forward latency. We use one NVIDIA A100-SXM4-80GB GPU with PyTorch 2.9.1 and CUDA 12.6. BF16 and HiRes share the GEMM and attention backends and mathematical precision settings; HiRes computes logical MXFP4 operations using decoded BF16 weights. Precompiled Block32 operators fuse transformed-input encoding, one-code refinement, and recovery with adjacent elementwise operations to reduce intermediate writes and kernel launches. Refinement reads compact G_Q directly and scores all 32×16 legal candidates, retaining FP32 scores, the negative-score acceptance rule, and tie-breaking conventions. Recovery retains its BF16 rounding boundaries.

The workload uses batch size 1, 2,048 input tokens, no generation, and `use_cache=False`, with weights and inputs already on the GPU. After 10 warm-up passes, we report the median of 30 CUDA-synchronized complete forwards. Timing includes input transforms, dynamic A4, candidate scoring and selection, compact-matrix access, structural recovery, GEMM, attention, and final logits. HiRes adds 3.97%–4.49% latency over BF16 (Table 29). Table 22 separately reports the one-code ablation under the eager/reference backend and its decoding workload.

One-time quantization. Construction uses C4-train seed 0 and exactly 128×2048 calibration tokens. Timing starts with BF16 weights and calibration data ready and ends with the quantized model and all auxiliary state constructed. It includes geometry, GPTQ, statistics recollection, recovery fitting, and inter-stage transfers. Table 29 reports absolute build times; BF16 has no quantization stage.

Table 29: Complete forward latency and one-time quantization time. Forward latency uses the fused implementation at batch size 1 and input length 2,048, with KV caching disabled. Quantization times cover the full HiRes construction.

Model	BF16 forward (ms)↓	HiRes forward (ms)↓	Increase	HiRes quantization (min)↓
Qwen3-8B	1286	1343	4.43%	61.6
Llama-3-8B	1224	1279	4.49%	57.8
Qwen3-32B	5438	5654	3.97%	286.9

D QuantForge: Additional Results

We first compare complete search systems under matched budgets. The following experiments isolate verdict-guided updates, control selection, and implementation checks using the same full-QuantForge reference runs.

D.1 Matched-Budget Comparisons

Score-only, TextMem, ReflectMem, and QuantForge use GPT-5.6 Sol with high reasoning effort and otherwise default settings. They share the initial RTN program, editable source, API constraints, evaluator, and eight paired seeds. As specified in Appendix A.1, Qwen3-4B supplies search feedback; a separate evaluator measures Llama-3.1-8B transfer without returning those results to search.

ReflectMem extends TextMem's free-form summaries with explicit comparison of successful and failed experiments. It records rejected directions and uses those comparisons to propose the next code change. It has the same access to visible-model diagnostics as QuantForge.

Reflection prompt and memory. All eight ReflectMem runs use the following instruction:

> Compare this round with its parent and the most relevant successful and failed experiments. Distinguish observations from causal inferences. Record rejected modifications and the conditions under which they were rejected. Propose one code change supported by these historical comparisons, and explain its expected benefit and possible failure modes. If revisiting a rejected direction, explain how the new conditions differ.

Memory retains the best candidate, the latest 20 experiments, up to 16 failed directions, and up to 16 summaries of successful mechanisms. The controller retrieves entries by relevance before assembling each request. When memory exceeds its allowance, it merges duplicates first and then removes the least relevant older entries, without preferentially removing failure records.

ReflectMem and QuantForge have the same application-level input budget of 65,536 tokens per proposer call, including at most 16,384 tokens of memory. The remaining capacity holds system instructions, current code, objectives, and diagnostics. Counts use the proposer tokenizer. QuantForge's residuals, explanations, controls, and verdict records all count toward the same memory allowance. Neither arm adds an application-level output cap; output settings retain the provider defaults. Context assembly is performed before each request.

Evaluator allocation. The 240-call search budget includes new candidates, controls, and compliance probes. Score-only, TextMem, and ReflectMem each evaluate 240 new candidates. QuantForge instead uses 141 calls for candidates, 69 for controls, and 30 for probes. ReflectMem combines reflection and code generation in one proposer call, followed by one candidate evaluation; its controls, compliance probes, and separate diagnostic reruns each consume zero calls. Reading stored diagnostics requires no additional evaluation.

A separate evaluator tests the champion on held-out Llama-3.1-8B every 20 calls, producing 12 probes per seed. These scores do not feed back to the proposer or candidate selection, and the probes are outside the 240-call search budget. Across eight seeds, ReflectMem uses 1,920 candidate evaluations, 96 held-out probes, and 1,920 proposer calls. The QuantForge reference is the same eight-run set used for the compilation and verdict-replay experiments in Appendix D.2.

Discovery outcomes. QuantForge has lower final perplexity than Score-only in eight paired seeds and than TextMem in seven. Table 30 summarizes these original three-arm runs; Table 32 includes ReflectMem.

ReflectMem reaches median PPL 8.09, compared with TextMem's 9.20 and QuantForge's 7.80 (Table 32). The mean paired difference, QuantForge minus ReflectMem, is -0.250000, with a paired $n = 8$ t interval of $[-0.386058, -0.113942]$. QuantForge improves in seven seeds; ReflectMem is better by 0.08 in seed 6. The $\mathrm{PPL} \leq 8.00$ gate is reached in $3/8$ ReflectMem runs and $6/8$ QuantForge runs.

Table 30: Transfer outcomes after 240 matched evaluator calls. Threshold cost identifies the median generation index of the first candidate confirmed to pass, among successful seeds.

Arm	Median PPL↓	Gate	Generation index↓	New programs
Score-only	10.03	1/8	226	240
TextMem	9.20	3/8	214	240
QuantForge	**7.80**	**6/8**	**177**	141

Threshold cost has two timestamps: the evaluator-call index at which the passing candidate was generated, and the later held-out call that confirmed it. Confirmation occurs every 20 calls, whereas generation can occur at any call. Table 31 lists both for successful seeds; the anytime curve uses confirmation calls and includes every seed.

Table 31: Recorded threshold crossings, restricted to successful seeds. Generation index identifies the candidate first confirmed below the threshold; confirmation occurs at a held-out observation.

Arm	Seed	Held-out PPL	Generation index	Confirmation call
QuantForge	4	7.38	138	140
QuantForge	6	7.47	152	160
QuantForge	8	7.59	175	180
QuantForge	1	7.74	179	180
QuantForge	7	7.86	208	220
QuantForge	3	7.98	233	240
TextMem	6	7.31	196	200
TextMem	4	7.51	214	220
TextMem	8	7.97	227	240
Score-only	4	7.92	226	240

The median generation/confirmation indices are $177/180$ for QuantForge, $214/220$ for TextMem, and $226/240$ for Score-only, conditioned on success. At calls 160, 200, and 240, QuantForge has 2, 4, and 6 confirmed successful seeds; Table 33 gives the median trajectories.

Table 32: Held-out Llama-3.1-8B WikiText-2 perplexity after 240 search calls. ReflectMem is the additional arm; the other three reuse the original paired runs. Negative paired differences favor QuantForge.

Seed	Score-only	TextMem	ReflectMem	QuantForge	QuantForge − ReflectMem
1	9.31	9.06	8.08	7.74	−0.34
2	12.83	11.85	9.73	9.31	−0.42
3	10.12	9.34	8.25	7.98	−0.27
4	7.92	7.51	7.53	7.38	−0.15
5	10.55	10.62	8.86	8.44	−0.42
6	9.60	7.31	7.39	7.47	+0.08
7	11.27	9.89	8.10	7.86	−0.24
8	9.94	7.97	7.83	7.59	−0.24
Median	10.03	9.20	8.09	7.80	—
Gate count	1/8	3/8	3/8	6/8	—

The BF16 reference is 6.24. We set the transfer gate to $\mathrm{PPL} \leq 8.00$ before running the comparison.

Table 33: Median held-out perplexity over the matched search budget.

Arm	40	80	120	160	200	240
Score-only	14.62	12.08	11.24	10.71	10.35	10.03
TextMem	14.55	11.72	10.58	9.94	9.51	9.20
ReflectMem	14.80	12.00	10.38	9.19	8.44	8.09
QuantForge	15.28	12.55	10.19	8.78	8.05	7.80

Discovered interventions. Table 34 classifies final champions from the original three arms by executable behavior: geometry appears frequently in every arm, while discrete realization and structural recovery are more frequent in QuantForge. Four QuantForge runs contain the full ordered hierarchy, compared with none in either baseline.

Table 34: Intervention-regime audit of final executable champions. Counts are behavioral matches out of eight runs, not textual mentions.

Regime	Score-only	TextMem	QuantForge
Geometric shaping	6/8	7/8	**8/8**
Discrete realization	1/8	3/8	**6/8**
Structural recovery	0/8	1/8	**4/8**
Full ordered hierarchy	0/8	0/8	**4/8**

D.2 Verdict-Guided Program Updates

These experiments separate the effect of requiring a code response from the information supplied by the experiment. The full-search ablations here and in Appendix D.3 share the protocol and eight QuantForge reference runs from Appendix D.1. Proposer settings and prompt templates are fixed; arms differ in their search-state content. Candidate generation calls the proposer, while controls and compliance probes call the evaluator directly. The local replay then compares successors from identical archived evidence.

Compiling conclusions into code. We remove mandatory implementation and compliance checks while retaining competing explanations, compiled controls, and result memory. The search controller, tool access, diagnostic interface, and information-access rules remain fixed. Each arm obtains evidence from its own successors.

All 30 released probe calls return to new-candidate generation: the ablated arm explores 171 candidates versus 141, or 21% more, with the corresponding 30 additional proposer calls (Table 35).

QuantForge still obtains lower held-out perplexity in $8/8$ paired seeds (Table 36). The mean paired difference is -0.275, SD is 0.111, and 95% CI is $[-0.368, -0.182]$.

Table 35: Per-run evaluator allocation with and without residual compilation. Every released probe call funds a new candidate.

Arm	New candidates	Controls	Probes	Total
QuantForge	141	69	30	240
Without compilation	171	69	0	240

Table 36: Paired held-out perplexity for the residual-compilation ablation. Differences are QuantForge minus the ablated arm.

Arm / seed	1	2	3	4	5	6	7	8
QuantForge	7.74	9.31	7.98	7.38	8.44	7.47	7.86	7.59
Without compilation	8.16	9.68	8.33	7.52	8.71	7.71	7.96	7.90
Paired difference	-0.42	-0.37	-0.35	-0.14	-0.27	-0.24	-0.10	-0.31

The median held-out PPL is 7.80 with residual compilation and 8.06 without it (Table 37). Verified residual closures count resolved questions with a confirmed executable response. Verified closures increase by $2.67\times$, while invalid successors fall from 0.38 to 0.13. The comparison measures mandatory implementation and compliance checking together. Appendix D.3 separates their contributions using a source-diff checker with the same 171-candidate budget as the ablated arm.

Table 37: Residual-compilation outcomes under the same total evaluator budget.

Metric	QuantForge	Without compilation
Median held-out PPL	7.80	8.06
Gate (≤ 8.00)	6/8	4/8
Median candidate generation index	177	216
Median confirmation call	180	220
Invalid-successor fraction	0.13	0.38
Verified residual closures per run	10.4 ± 2.6	3.9 ± 1.5

Replaying an experimental verdict. A verdict constraint improves successors even when both arms receive literally identical evidence. Each replay fixes the parent program, history, and control result; one successor uses free reflection, and the other must implement the verdict. We specified the following sampling rule before running the replay: take each trajectory's first 6 controls that exclude at least one registered competing explanation, ordered by evaluator call. All eight trajectories contain at least 6 such controls, averaging 11.3. This gives $8 \times 6 = 48$ transitions, with 26 supported and 22 refuted explanations. We generate and evaluate both successors during replay, regardless of whether the original search expanded that parent.

The improving-successor rate rises from $15/48 = 0.31$ to $28/48 = 0.58$, and probe-confirmed implementation rises from 0.35 to 0.92 (Table 38). The gain is larger after refutation, from 0.18 to 0.45, than after support, from 0.42 to 0.69 (Table 39). These correspond to relative gains of 2.5 and 1.6, respectively. A supported explanation already invites continuation; a refuted one requires the successor to change direction. All eight trajectory-level paired differences are positive (Table 40), with mean $+1.625$ and SD 0.518.

Table 38: Successor outcomes from replaying the same evidence under free reflection or a verdict constraint.

Metric	Free reflection	Verdict-constrained
Improving successors	15/48 = 0.31	28/48 = 0.58
Mean visible PPL change	-0.04	-0.21
Probe-confirmed implementation	0.35	0.92
Repeated refuted mechanism	0.29	0.08

Table 39: Improving successors by verdict type in the paired replay.

Verdict	Count	Free reflection	Verdict-constrained
Explanation supported	26	11/26 = 0.42	18/26 = 0.69
Explanation refuted	22	4/22 = 0.18	10/22 = 0.45

Table 40: Improving successors by source trajectory, with 6 replayed transitions per trajectory.

Condition / seed	1	2	3	4	5	6	7	8
Free reflection	2	1	2	3	1	2	2	2
Verdict-constrained	4	3	4	4	3	3	4	3
Paired difference	+2	+2	+2	+1	+2	+1	+2	+1

An example of a verdict-guided update. On visible Qwen3-4B, A4 input-quantization MSE for `down_proj` in layers 18–23 is $3.1\times$ that of `up_proj` inputs in the same layers. The 9728 input features form 304 physical Block32 groups per row.

Two explanations were registered. (a) *Channel alignment*: a few SwiGLU output channels have much larger magnitudes and occupy adjacent positions. Their concentration in a few blocks raises shared exponents and causes other E2M1 elements in those blocks to underflow. (b) *Global dynamic range*: the tensor has a large overall range, with error distributed roughly uniformly across its 304 blocks; block membership is irrelevant.

The control fixes a random permutation P of the `down_proj` input features and inversely reorders the weight columns:

$$Wx = (WP^\top)(Px).$$

This diagnostic control preserves the network function, scale rule, block size, and shared-exponent format while changing block membership. Explanation (a) predicts that dispersing large channels contaminates more blocks and raises MSE. Explanation (b) predicts approximately unchanged MSE.

The error ratio rises from $3.1\times$ to $4.6\times$. Contaminated blocks, defined by a within-block maximum-to-median absolute-magnitude ratio > 32, increase from 7.2% to 41.5%. This verdict rejects (b) and supports (a) for the registered comparison. The resulting evidence entry records the dependence on block membership for `down_proj` inputs in layers 18–23.

The verdict-constrained successor retains the fixed MXFP4 block layout. Instead, `equalize_channels()` ranks channels by their contribution to their block's shared exponent, replacing global magnitude ranking. It rescales only the highest contributors, dividing activations by s and multiplying the corresponding `down_proj` weight columns by s. This moves magnitude to the weight side while preserving the function. Reverting the selection criterion changes the probe output, confirming implementation. Contaminated blocks fall from 7.2% to 2.4%, and visible PPL falls from 11.42 to 11.19.

The free-reflection successor instead widens the per-block scale-search range on the same parent, continuing along explanation (b). Its visible PPL is 11.44, with no improvement. The two successors illustrate how the same evidence leads to different code changes under reflection and a required verdict response.

D.3 Control Selection and Implementation Checks

Control selection. We compare the current minimal discriminating policy with equal-cost random legal controls and execution of all pre-specified controls. Every control consumes the same 240-call budget used for candidates and probes. An *explicit verdict* excludes at least one registered explanation, irrespective of successor behavior. An *effective verdict* also reaches the successor's executed computation, as confirmed by a probe. The latter is the verified-closure count used in Appendix D.2.

Minimal discriminating selection gives the best budget-limited result among the three policies (Table 41). It improves held-out PPL over both alternatives in $8/8$ seeds (Table 42). At the same 69 control calls, random selection yields 4.6 explicit verdicts versus 11.3, and costs 15.0 calls per verdict versus 6.1. This comparison measures discrimination before implementation enters the count. Executing all registered controls yields 12.1 explicit verdicts, but its 118 control calls leave only 92 new candidates rather than 141. Median held-out PPL worsens to 8.02 despite the additional verdicts.

Table 41: Control-selection ablation under the 240-call budget. Explicit verdicts exclude an explanation; effective verdicts also reach code, as confirmed by a probe.

Policy	Control calls	New candidates	Explicit verdicts	Calls per explicit verdict	Effective verdicts	Median PPL	Gate
Minimal discriminating	69	141	11.3	6.1	10.4	7.80	6/8
Random legal	69	141	4.6	15.0	4.2	8.31	3/8
All pre-specified	118	92	12.1	9.8	11.1	8.02	4/8

Table 42: Held-out perplexity by seed for the control-selection policies.

Policy / seed	1	2	3	4	5	6	7	8
Minimal discriminating	7.74	9.31	7.98	7.38	8.44	7.47	7.86	7.59
Random legal	8.29	9.74	8.60	7.61	8.88	7.83	8.33	7.94
All pre-specified	7.88	9.52	8.19	7.55	8.63	7.72	8.06	7.98

Implementation rates are similar across policies: $10.4/11.3$, $4.2/4.6$, and $11.1/12.1$, or 0.91–0.92. The two comparisons separate control quality from budget allocation. Equal-cost random controls yield less discriminating evidence; running all controls trades candidate exploration for more verdicts.

Implementation checks. We compare source diffs, the current execution probe, and strict layer-output comparison. The same 40 injected cases contain 20 genuine implementations and 20 apparent edits that leave the target computation unchanged. Two blinded annotators agree on $38/40$ reference labels; discussion resolves the remaining 2 cases. Each checker is tested on these cases and in the eight-seed search comparison.

False accepts count apparent edits accepted as implemented, out of 20. False rejects count genuine implementations rejected, also out of 20. Rejected successors must be redone and consume candidate budget. Table 43 reports both errors, checking calls, remaining candidates, and held-out performance. The execution probe gives the lowest median held-out PPL among the three configurations.

Table 43: Implementation-checker errors on 40 injected cases and subsequent search outcomes. False accepts and false rejects each use 20 cases as the denominator.

Checker	False accepts	False rejects	Extra calls	New candidates	Median PPL	Gate
Source diff	0.45 (9/20)	0.00 (0/20)	0	171	7.96	4/8
Execution probe	0.05 (1/20)	0.05 (1/20)	30	141	7.80	6/8
Layer-output comparison	0.00 (0/20)	0.20 (4/20)	74	97	7.91	5/8

The source-diff configuration retains the implementation requirement but costs no extra evaluator calls. It shares the 171-candidate budget of the no-compilation arm. The change from 8.06 to 7.96 therefore separates the implementation requirement under source-diff checking from removing that requirement. Replacing source diffs with execution probes lowers the median to 7.80. The sequence $8.06 \to 7.96 \to 7.80$ records the combined gain as 0.10 from the requirement under source checking and 0.16 from the execution-probe configuration. Source diffs accept 0.45 of apparent implementations, consistent with evidence entries closing while the targeted residual persists.

Stricter layer-output comparison accepts none of the 20 apparent edits, but rejects $4/20$ genuine implementations. Its 74 additional calls and rejected successors leave 97 new candidates, versus 141 for execution probes; median PPL rises to 7.91. The false-accept count is $0/20$, with a 95% upper confidence bound of 0.17.