TReQ: Refitting Output Transformations Without Requantizing Language Models

Anonymous authors
Paper under double-blind review

Abstract

Low-bit weight–activation quantization introduces errors that can accumulate across language-model layers. We present TReQ, a post-training quantization method with block-diagonal transformations before and after low-precision matrix multiplication. The output transformations mix channels within small blocks, providing additional reconstruction freedom while keeping quantized weights and scales fixed during each refit. We combine two-sided transformation construction with reconstruction of attention scores, attention context, and downstream projection outputs. The output transformations can be fused into the GEMM epilogue, avoiding a separate kernel and intermediate-memory round trip. Across Ministral-3-8B, Llama-3.1-8B, and Qwen3-8B, TReQ reduces block reconstruction error relative to matched WUSH baselines in INT4, MXFP4, and NVFP4. Under INT4, it lowers task Kullback–Leibler (KL) divergence and improves accuracy on all five benchmarks for every model, while reducing WikiText-2 perplexity. End-to-end improvements in the FP4 formats are less consistent. A fixed-checkpoint Ministral INT4 comparison shows that output refitting alone reduces held-out KL by 8.63%. On NVIDIA B200, fused output transformations add only 0.1–1.5% to MXFP4 full-model prefill latency at the tested workloads with at least 2,048 total tokens.

1 Introduction

Post-training quantization reduces the memory traffic required by large language models without repeating pretraining. Weight quantization methods such as GPTQ (Frantar et al., 2023) and AWQ Lin et al. (2024) use calibration data to preserve model quality at reduced precision. Quantizing activations as well as weights allows the dominant matrix multiplications to use low-precision operands, but introduces a second source of approximation error. Large activation outliers and uneven channel magnitudes make this setting particularly challenging (Dettmers et al., 2022; Xiao et al., 2023; Ashkboos et al., 2024).

One way to reduce these errors is to transform activations and weights before quantization. SmoothQuant rescales channels (Xiao et al., 2023); QuaRot and SpinQuant use fixed or learned rotations (Ashkboos et al., 2024; Liu et al., 2025); FlatQuant learns affine transformations (Sun et al., 2025); and WUSH computes small block-diagonal transformations from activation and weight second moments (Chen et al., 2026). These approaches improve the values presented to the quantizer. We study whether fitting transformations after the low-precision multiplication can correct the model's outputs while keeping the quantized weights and their scales fixed.

TReQ retains small matrices after low-precision matrix multiplication as adjustable *output transformations*. Each block forms linear combinations of neighboring output channels, expanding the reconstruction choices beyond independent channel scaling. We combine a two-sided construction for layer quantization with calibration-based output refitting. During each refit, weight codes and quantization scales stay fixed; the adjustable coefficients act directly on the existing low-precision product.

We extend this reconstruction to the computations that couple transformer projections. Query and key errors interact in attention scores, and attention probabilities mix value errors into the context; independent projection fitting can miss these interactions. Motivated by attention-aware quantization

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.

(Kim et al., 2025; 2026), we refit output transformations to reconstruct attention scores, attention context, and downstream projection outputs from the original bfloat16 (BF16) model.

The block-diagonal structure also makes output transformations inexpensive to execute. Each small transformation operates on a block of GEMM output while it remains on chip, allowing fusion into the epilogue without a separate kernel or a round trip through global memory. Our native MXFP4 implementation on NVIDIA B200 adds only 0.1–1.5% to full-model prefill latency at the tested workloads with at least 2,048 total tokens.

We evaluate TReQ against matched WUSH baselines on Ministral-3-8B, Llama-3.1-8B-Instruct, and Qwen3-8B in INT4, MXFP4, and NVFP4, with four-bit weights and activations. TReQ reduces mean held-out block reconstruction error in all nine model–format settings. Under INT4, it lowers task Kullback–Leibler (KL) divergence and improves accuracy on all five benchmarks for each model, alongside lower WikiText-2 perplexity. The FP4 formats show less consistent end-to-end improvements. A fixed-checkpoint Ministral INT4 comparison further shows that output refitting alone reduces held-out KL by 8.63%, with weight codes, scales, and input transformations held fixed.

2 Related Work

Calibration and reconstruction. Optimal Brain Compression and GPTQ use second-order information to compensate for sequential quantization errors (Frantar & Alistarh, 2022; Frantar et al., 2023). AWQ uses activation statistics to select channel scaling that protects salient weights (Lin et al., 2024). AdaRound optimizes weight rounding through layer reconstruction (Nagel et al., 2020); BRECQ extends reconstruction to blocks (Li et al., 2021). OmniQuant minimizes this reconstruction error by learning weight clipping and equivalent transformations (Shao et al., 2024), while TesseraQ optimizes rounding decisions and dequantization scales through progressive adaptive rounding and block reconstruction (Li & Panda, 2024). Our method combines input transformations derived from second-moment statistics with reconstruction-based fitting of small output transformations.

Scaling, rotations, and adaptive transformations. SmoothQuant redistributes magnitudes between activations and weights through channel rescaling (Xiao et al., 2023). QuIP and QuIP# use incoherence processing for low-bit weight quantization (Chee et al., 2023; Tseng et al., 2024). QuaRot uses rotations that preserve the original full-precision computation to suppress outliers (Ashkboos et al., 2024), and SpinQuant learns rotations to improve quantized network accuracy (Liu et al., 2025). FlatQuant learns affine transformations with Kronecker structure to reduce their execution cost (Sun et al., 2025). WUSH derives non-orthogonal block-diagonal transformations from second moments and a Hadamard matrix (Chen et al., 2026). We build on WUSH for initialization and for a short search over input transformations. Our output transformations remain adjustable after quantization, allowing subsequent refits without changing weight codes or scales. Section 3.2 describes how the transformed weight is derived at each construction step. For microscaling FP4, MR-GPTQ combines blockwise Hadamard transforms with format-specific quantization optimizations (Egiazarian et al., 2026).

Attention-aware quantization and error propagation. BoA constructs attention-aware Hessians to account for interactions among projections during backpropagation-free weight quantization (Kim et al., 2025). TurboBoA quantizes multiple output channels jointly and corrects errors from preceding quantized layers (Kim et al., 2026). Its attention-wise refinement freezes integer weight codes and refits per-output-channel scales, equivalent to a diagonal output transformation, through closed-form coordinate updates that reduce attention reconstruction error. Our block-diagonal output transformations additionally mix channels within each block, while keeping the original weight codes and quantization scales fixed. Section 3.1 describes this additional reconstruction freedom.

Adapting frozen quantized models. QLoRA trains LoRA adapters on a frozen quantized backbone (Hu et al., 2022; Dettmers et al., 2023); LQ-LoRA decomposes weights into quantized and trainable low-rank components (Guo et al., 2024). PEQA instead updates quantization scales (Kim et al., 2023). We fit block-diagonal output transformations while keeping weight codes and scales fixed. These transformations act on the result of the existing low-precision matrix multiplication, so

they need no separate copy of the layer input. We fit them to projection outputs, attention scores, and attention context from the original BF16 model, without task labels.

3 METHOD

TReQ places block-diagonal transformations before and after the low-precision matrix multiplication. We fit the output transformations to match the full-precision computation as accurately as possible. The output transformations improve quantization accuracy because they linearly combine the quantized weights within each block: the effective weights can approximate any point in the subspace the discrete quantized weights span. On the other hand, the block-diagonal transformations are also hardware-friendly: they can be fused into the GEMM epilogue, introducing little overhead in practice (0.1–1.5% added full-model prefill latency with MXFP4 on B200; Section 4.5). Section 3.1 defines this quantized linear layer and explains the expressive capacity and execution cost of its output transformations. Section 3.2 describes how to construct the layer through transformation initialization, input and output updates, and final weight quantization. Section 3.3 extends reconstruction from individual linear layers to attention scores, attention context, and downstream attention and MLP projection outputs.

3.1 THE TREQ LAYER

Consider a linear layer $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{W}_0$ (bias omitted) with input activations $\boldsymbol{A} \in \mathbb{R}^{n \times d_{\text{in}}}$ for n tokens and pretrained weight $\boldsymbol{W}_0 \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$. A TReQ layer replaces $\boldsymbol{W}_0$ with a quantized weight $\widehat{\boldsymbol{W}} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, whose entries are values representable in a low-precision format such as MXFP4 or INT4. It surrounds $\widehat{\boldsymbol{W}}$ with an input transformation $\boldsymbol{T}_{\text{in}} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$ and an output transformation $\boldsymbol{T}_{\text{out}} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}$, both block diagonal with block size b, and approximates the layer output by

$$\boldsymbol{Y} \approx \widehat{\boldsymbol{Y}} = \underbrace{\mathcal{Q}_A(\boldsymbol{A}\boldsymbol{T}_{\text{in}})\,\widehat{\boldsymbol{W}}}_{\boldsymbol{Z}}\,\boldsymbol{T}_{\text{out}}, \tag{1}$$

where the activation quantizer $\mathcal{Q}_A$ likewise quantizes $\boldsymbol{A}\boldsymbol{T}_{\text{in}}$ to values representable in a low-precision format. The *GEMM output* $\boldsymbol{Z}$ is thus an ordinary matrix product, which the hardware computes natively in these formats. In practice, $\widehat{\boldsymbol{W}}$ and the quantized activations are low-bit (e.g., 4-bit), whereas both transformations are kept in 16-bit or higher precision, so their entries are effectively continuous.

Expressiveness. Assume that b divides d_{out}, and let $\boldsymbol{Z}_j \in \mathbb{R}^{n \times b}$ and $\boldsymbol{T}_{\text{out},j} \in \mathbb{R}^{b \times b}$ be the j-th b-channel block of $\boldsymbol{Z}$ and the j-th block of $\boldsymbol{T}_{\text{out}}$. The corresponding output block is $\boldsymbol{Z}_j\boldsymbol{T}_{\text{out},j} \in \mathbb{R}^{n \times b}$, so each output channel is a linear combination of the b columns of $\boldsymbol{Z}_j$. Since the coefficients are effectively continuous, each output channel can be any vector in the span of these columns. The layer thus represents a whole subspace rather than a discrete set of quantized values, which leaves room to reduce quantization error.

Efficiency. Because $\boldsymbol{T}_{\text{out}}$ is block diagonal, each block multiplies a b-channel block of $\boldsymbol{Z}$ while it is still on chip, before the final store. $\boldsymbol{T}_{\text{out}}$ is therefore fused into the GEMM epilogue, which avoids a separate kernel launch and a round trip of $\boldsymbol{Z}$ through global memory. The output transformation requires approximately $2nbd_{\text{out}}$ FLOPs, compared with $2nd_{\text{in}}d_{\text{out}}$ for the main GEMM, an arithmetic overhead of b/d_{in}. For example, at $b = 32$ and $d_{\text{in}} = 4096$, the added FLOPs are only 0.78%. Accounting for the different arithmetic precisions, B200's dense Tensor Core peaks of 9 PFLOPS for FP4 and 1.125 PFLOPS for TF32 (NVIDIA) give an estimated compute-time ratio of $(32/4096)(9/1.125) = 6.25\%$ for a TF32 output transformation. This estimate assumes both operations reach their respective peaks; lower utilization of the main GEMM relative to the epilogue can make the actual overhead smaller. The output transformation can also use FP16, which has the same mantissa precision as TF32 within the FP16 normal range. Section 4.5 evaluates both precisions and demonstrates lower overhead in measured GEMM examples and reports just 0.2–1.5% added full-model prefill latency with TF32 output transformations for the tested workloads with at least 2,048 total tokens.

For clarity, we describe TReQ for a single weight matrix. In practice, some projections are split into a few groups of input and output channels, aligned with the b-channel blocks and attention heads.

Each input/output group pair is a TReQ layer on its slice of the weight, and the outputs of the input groups are summed.

3.2 Optimizing a TReQ layer

Given calibration inputs $\boldsymbol{A}$, we seek $\boldsymbol{T}_{\text{in}}$, $\widehat{\boldsymbol{W}}$, and $\boldsymbol{T}_{\text{out}}$ such that Eq. (1) approximates $\boldsymbol{Y}$. Optimizing them jointly is hard: $\widehat{\boldsymbol{W}}$ is discrete, and $\boldsymbol{T}_{\text{in}}$ acts inside the activation quantizer $\mathcal{Q}_A$. Only $\boldsymbol{T}_{\text{out}}$ enters Eq. (1) linearly. We therefore alternate between two updates. With $\boldsymbol{T}_{\text{in}}$ and $\widehat{\boldsymbol{W}}$ fixed, we fit $\boldsymbol{T}_{\text{out}}$ by regularized least squares. With $\boldsymbol{T}_{\text{out}}$ fixed, we construct $\boldsymbol{T}_{\text{in}}$ from second-moment statistics and quantize the corresponding transformed weight to obtain $\widehat{\boldsymbol{W}}$.

We use the normalized input second-moment matrix $\boldsymbol{H}_{\text{in}} = d_{\text{in}} \boldsymbol{A}_c^\top \boldsymbol{A}_c / \operatorname{tr}(\boldsymbol{A}_c^\top \boldsymbol{A}_c)$, computed from calibration inputs $\boldsymbol{A}_c$ of this layer in the original BF16 model and held fixed throughout construction. It estimates $\boldsymbol{A}^\top \boldsymbol{A}$ up to scale. We initialize both transformations using the two-sided WUSH construction (Appendix C).

Optimizing $\boldsymbol{T}_{\text{out}}$ with $\boldsymbol{T}_{\text{in}}$ and $\widehat{\boldsymbol{W}}$ fixed. With $\boldsymbol{T}_{\text{in}}$ and $\widehat{\boldsymbol{W}}$ fixed, the GEMM output $\boldsymbol{Z}$, including its activation quantization error, is fixed, and the layer output $\boldsymbol{Z}\boldsymbol{T}_{\text{out}}$ is linear in $\boldsymbol{T}_{\text{out}}$. Let $\boldsymbol{Y}_{\text{ref}}$ be the original BF16 layer output on these inputs and $E_Y = \|\boldsymbol{Y}_{\text{ref}}\|_F^2$. For $\lambda \in [0, 1]$, we form a continuous fitting proposal by minimizing

$$\frac{1-\lambda}{\kappa_r} \frac{\|\boldsymbol{Z}\boldsymbol{T}_{\text{out}} - \boldsymbol{Y}\|_F^2}{E_Y} + \frac{\lambda}{\kappa_w} \frac{\|\boldsymbol{T}_{\text{in}} \widehat{\boldsymbol{W}} \boldsymbol{T}_{\text{out}} - \boldsymbol{W}_0\|_F^2}{\|\boldsymbol{W}_0\|_F^2}. \tag{2}$$

The first term reconstructs the layer output on the calibration inputs. The second, the *effective-weight penalty*, keeps the effective weight $\boldsymbol{T}_{\text{in}} \widehat{\boldsymbol{W}} \boldsymbol{T}_{\text{out}}$ close to $\boldsymbol{W}_0$ and thus regularizes the fit. The constants κ_r and κ_w are the mean diagonal entries of the Gram matrices of the two normalized terms, so that λ sets their relative curvature. Since $\boldsymbol{T}_{\text{out}}$ is block diagonal, Eq. (2) separates into one regularized least-squares problem per $b \times b$ block, which we solve in closed form with a small ridge for numerical stability. Let $\boldsymbol{T}_{\text{out}}^*$ be the solution. It ignores inference-time rounding (e.g., of $\boldsymbol{T}_{\text{out}}$ to TF32 and of the layer output), so we perform a line search over

$$\boldsymbol{T}_{\text{out}}(\alpha) = \boldsymbol{T}_{\text{out}} + \alpha\big(\boldsymbol{T}_{\text{out}}^* - \boldsymbol{T}_{\text{out}}\big), \qquad \alpha \in \{0, 1, \tfrac{1}{2}, \tfrac{1}{4}, \tfrac{1}{8}\}.$$

Each candidate is evaluated with $\mathcal{D}$, the computation of Eq. (1) with all inference-time rounding applied: we substitute its output for $\boldsymbol{Z}\boldsymbol{T}_{\text{out}}$ and the BF16 reference $\boldsymbol{Y}_{\text{ref}}$ for $\boldsymbol{Y}$ in Eq. (2), keeping the candidate with the lowest value. Since $\alpha = 0$ keeps the current $\boldsymbol{T}_{\text{out}}$, the update never increases this rounded objective.

Optimizing $\widehat{\boldsymbol{W}}$ and $\boldsymbol{T}_{\text{in}}$ with $\boldsymbol{T}_{\text{out}}$ fixed. We absorb the current output transformation into the pretrained weight to form the *input-update target*

$$\boldsymbol{W}_{\text{target}} = \boldsymbol{W}_0 \boldsymbol{T}_{\text{out}}^{-1}. \tag{3}$$

Applying WUSH (Chen et al., 2026) to $\boldsymbol{H}_{\text{in}}$ and $\boldsymbol{W}_{\text{target}}$ constructs a new input transformation $\boldsymbol{T}_{\text{in}}$ from the input and weight second moments. The corresponding transformed weight is

$$\boldsymbol{W}_{\text{mid}} = \boldsymbol{T}_{\text{in}}^{-1} \boldsymbol{W}_{\text{target}} = \boldsymbol{T}_{\text{in}}^{-1} \boldsymbol{W}_0 \boldsymbol{T}_{\text{out}}^{-1}. \tag{4}$$

We then set $\widehat{\boldsymbol{W}} = \mathrm{RTN}(\boldsymbol{W}_{\text{mid}})$, where RTN rounds each entry to the nearest representable value.

Alternating optimization. Starting from the two-sided WUSH initialization, we quantize $\boldsymbol{W}_{\text{mid}}$ with RTN, fit $\boldsymbol{T}_{\text{out}}$, and save the initial state. We then repeat the input/weight update followed by the output-transformation fit, saving a state after each fit. After several rounds, we select the saved RTN *candidate state* with the lowest relative output error on the fitting inputs, evaluated with inference-time rounding,

$$L_{\text{deploy}} = \frac{\|\mathcal{D}(\boldsymbol{A}; \boldsymbol{T}_{\text{in}}, \widehat{\boldsymbol{W}}, \boldsymbol{T}_{\text{out}}) - \boldsymbol{Y}_{\text{ref}}\|_F^2}{E_Y}. \tag{5}$$

Final quantization. For the selected RTN candidate, we reconstruct $\boldsymbol{W}_{\mathrm{mid}}$ using Eq. (4) and quantize it once with a two-sided extension of GPTQ (Frantar et al., 2023). With the transformations fixed, this pass compensates weight-quantization errors using input and output metrics. Specifically, $\mathrm{GPTQ}(\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{G})$ takes a continuous weight $\boldsymbol{W}$ and two positive semidefinite matrices $\boldsymbol{H}$ and $\boldsymbol{G}$ and returns a quantized weight $\widehat{\boldsymbol{W}}$ that approximately minimizes $\mathrm{tr}\big((\widehat{\boldsymbol{W}} - \boldsymbol{W})^\top \boldsymbol{H} (\widehat{\boldsymbol{W}} - \boldsymbol{W}) \boldsymbol{G}\big)$. We compute

$$\widehat{\boldsymbol{W}} = \mathrm{GPTQ}\big(\boldsymbol{W}_{\mathrm{mid}},\ \boldsymbol{T}_{\mathrm{in}}^\top \boldsymbol{H}_{\mathrm{in}} \boldsymbol{T}_{\mathrm{in}},\ \boldsymbol{T}_{\mathrm{out}} \boldsymbol{T}_{\mathrm{out}}^\top\big).$$

We finally refit $\boldsymbol{T}_{\mathrm{out}}$ with the resulting weight codes and scales fixed. Appendix A.2 specifies the two-sided error-compensation procedure.

3.3 Adapting TReQ to transformer blocks

Each attention and MLP projection is a quantized linear layer. Single-layer reconstruction addresses the local error of each projection; we next refit output transformations to reconstruct the larger computations in which these projections participate.

Attention couples different projections across tokens. Query and key errors jointly perturb their dot products: the score error depends on each error's alignment with the other projection, not just its magnitude. After softmax, changed attention probabilities mix with value errors to perturb the context. Independent projection MSE therefore does not directly target either composed computation. We refit Q/K against BF16 scores and V against BF16 context, then refit O and MLP down against their original outputs to account for propagated errors. Throughout these refits, only output transformations are adjustable; input transformations, weight codes and scales, and the activation-quantizer configuration remain fixed.

Query and key projections. We refit the Q/K output transformations to reconstruct the original BF16 attention scores before softmax. For one query head and its associated key head, let $\boldsymbol{q}_t^*$ and $\boldsymbol{k}_u^*$ be the original BF16 queries and keys after any head normalization and RoPE, and $\widetilde{\boldsymbol{q}}_t$ and $\widetilde{\boldsymbol{k}}_u$ their quantized-model counterparts. Let $\boldsymbol{R}_t$ be the RoPE matrix at position t and d_{head} the head dimension. The reference scores and reconstruction loss are

$$S_{tu}^* = \frac{\boldsymbol{q}_t^{*\top} \boldsymbol{k}_u^*}{\sqrt{d_{\mathrm{head}}}}, \qquad \mathcal{L}_{\mathrm{score}} = \frac{\sum_{u \le t} (\widetilde{S}_{tu} - S_{tu}^*)^2}{\sum_{u \le t} (S_{tu}^*)^2},$$

where $\widetilde{S}$ denotes the scores computed from $\widetilde{\boldsymbol{q}}_t$ and $\widetilde{\boldsymbol{k}}_u$. The loss sums over all sequences and query heads.

With K fixed, we update Q's output transformation by $\Delta \boldsymbol{T}_{\mathrm{out}}^Q$. Let $\Delta \boldsymbol{y}_t^Q$ be the resulting change in the Q projection output before arithmetic rounding, and define $\Delta \boldsymbol{q}_t = \boldsymbol{R}_t \boldsymbol{J}_t \Delta \boldsymbol{y}_t^Q$ as its local change after normalization and RoPE. Here $\boldsymbol{J}_t$ is the head-normalization Jacobian at the current query, or the identity when no normalization is used. We fit the update using the score-reconstruction term

$$\frac{\sum_{u \le t} \left(\widetilde{S}_{tu} - S_{tu}^* + \frac{\Delta \boldsymbol{q}_t^\top \widetilde{\boldsymbol{k}}_u}{\sqrt{d_{\mathrm{head}}}} \right)^2}{\sum_{u \le t} (S_{tu}^*)^2},$$

together with the effective-weight penalty. Here $\widetilde{S}_{tu} - S_{tu}^*$ is the current quantized-model score error, and the remaining term is its first-order change before arithmetic rounding. Since $\Delta \boldsymbol{q}_t$ is linear in $\Delta \boldsymbol{T}_{\mathrm{out}}^Q$, this local objective is quadratic; with Q/K RMSNorm, it gives a Gauss–Newton update. We alternate this Q update with the corresponding K update, holding the accepted Q fixed. For Qwen3, we first refit the normalized Q/K projection outputs before this score refinement.

Value projection. Using the attention probabilities from the refitted Q/K projections, we fit V's output transformation to reconstruct the original BF16 attention context. The fitting prediction is the probability-weighted sum of V outputs without arithmetic rounding; the target uses the original BF16 probabilities and values. We minimize their squared difference with an effective-weight penalty against the pretrained V weight. Line-search candidates are evaluated with $\mathcal{D}$ for V, followed by the original BF16 attention output projection, O. We accept the update only if it reduces internal-validation reconstruction error.

Downstream attention and MLP projections. We then quantize O and the MLP's gate, up, and down projections in execution order, recomputing activations after each replacement. Their single-layer fits reconstruct the outputs obtained by applying the original weights to the current inputs. Gate and up retain these fits. For O and down, we subsequently refit the output transformations against fixed BF16 reference outputs captured before any projection in the block is replaced and before residual addition. These refits keep quantized weights and scales fixed and require internal-validation improvement.

4 EXPERIMENTS

4.1 SETUP AND METRICS

We compare TReQ with matched native WUSH adaptations on Ministral-3-8B, Llama-3.1-8B-Instruct, and Qwen3-8B in INT4, MXFP4, and NVFP4 (A4W4). Within each model, methods share Tülu input-second-moment captures and model-native evaluation protocols. We use disjoint sets of Tülu conversations for Hessian estimation, reconstruction fitting, MSE-based validation, and held-out evaluation, with 32 conversations reserved for the held-out set. Full-model fidelity is token-averaged KL from BF16 on shared BF16-generated histories; task KL uses 64 prompts per benchmark, whereas accuracy uses complete benchmark sets. WikiText-2 PPL is measured separately on raw text. Model-quality measurements use numerical quantize/dequantize; native-kernel timings are reported separately.

4.2 HELD-OUT BLOCK RECONSTRUCTION

We evaluate all 34 Ministral, 32 Llama, and 36 Qwen3 transformer blocks on the 32 held-out conversations. Both methods receive the same block inputs from the original BF16 model and are evaluated against its complete block outputs, including residuals. MSE is computed over all output elements. Table 1 reports the equally weighted mean of per-block changes, $100(\mathrm{MSE}_{\mathrm{TReQ}}/\mathrm{MSE}_{\mathrm{WUSH}} - 1)$. TReQ reduces MSE in 102/102 INT4, 97/102 MXFP4, and 100/102 NVFP4 comparisons.

Table 1: All-block held-out MSE change from WUSH to TReQ (A4W4). Cells give mean percentage change and improved blocks / total. Negative values indicate lower MSE; every block enters the mean.

Model	INT4	MXFP4	NVFP4
Ministral-3-8B	-37.17% (34/34)	-21.59% (34/34)	-23.19% (34/34)
Llama-3.1-8B	-55.43% (32/32)	-7.06% (30/32)	-6.06% (32/32)
Qwen3-8B	-32.22% (36/36)	-6.61% (33/36)	-9.43% (34/36)

4.3 MATCHED FULL-PIPELINE COMPARISONS

Table 2 presents the main results across three models and three quantization formats. For INT4, TReQ lowers task KL and improves zero-shot accuracy on all five benchmarks for every model, while also reducing WikiText-2 PPL. For MXFP4 and NVFP4, Ministral achieves lower task KL on all five benchmarks, although accuracy gains are less consistent. The Llama and Qwen3 results show a mixture of improvements and regressions in both KL and accuracy. Together with the block reconstruction results in Section 4.2, these findings show that lower average block MSE does not consistently translate into lower full-model KL or higher zero-shot accuracy.

4.4 OUTPUT REFITTING WITH QUANTIZED WEIGHTS FIXED

To isolate output refitting, we add identity output transformations to one Ministral INT4 WUSH checkpoint and compare leaving them unchanged (OFF) with fitting them (ON). Codes, scales, input transformations, grouping, and the augmented execution path are identical. Refitting reduces held-out KL by 8.63% (paired 95% CI: 7.38–9.90%) and fitting-set KL by 10.85% (9.12–12.56%).

Table 2: Headline results under matched Tülu calibration for each model. WikiText-2 PPL uses real text; task KL uses 64 BF16-trajectory prompts per task, while accuracy uses each complete benchmark and is reported as a percentage. The shared BF16 column gives the full-precision reference. Parentheses after each method name give bits per weight entry (BPE), including scales and transformations. Bold marks the better point estimate within each model and codec. Lower is better for PPL/KL; higher for accuracy.

Ministral-3-8B-Instruct													
	INT4				MXFP4				NVFP4				BF16
	WUSH (4.110)		TReQ (4.356)		WUSH (4.341)		TReQ (4.507)		WUSH (4.591)		TReQ (4.757)		
WikiText-2 PPL ↓	14.352		**11.003**		7.592		**7.506**		7.310		**7.268**		6.913
Dataset	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	Acc. ↑
MATH	.1975	38.74	**.1772**	**45.36**	.03333	66.06	**.03281**	**68.82**	.02088	66.14	**.01794**	**68.44**	66.18
MMLU	.4792	51.42	**.3534**	**61.79**	.09507	73.10	**.07464**	**73.68**	.06088	74.20	**.02967**	**74.62**	76.46
IFEval	1.0404	32.53	**.7135**	**45.84**	.1664	57.30	**.1546**	**58.96**	.11417	56.38	**.09872**	**56.56**	57.86
HumanEval	.2118	42.68	**.1085**	**56.10**	.02410	**77.44**	**.02031**	**77.44**	.01666	**80.49**	**.01297**	75.00	76.83
HellaSwag	.1708	63.11	**.1216**	**69.02**	.02201	78.00	**.02199**	**78.11**	.01363	78.91	**.01231**	**78.98**	80.58
Llama-3.1-8B-Instruct													
	INT4				MXFP4				NVFP4				BF16
	WUSH (4.110)		TReQ (4.356)		WUSH (4.341)		TReQ (4.507)		WUSH (4.591)		TReQ (4.757)		
WikiText-2 PPL ↓	25.756		**12.710**		**8.116**		8.162		**7.760**		7.804		7.217
Dataset	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	Acc. ↑
MATH	.3738	4.96	**.1583**	**19.12**	**.03534**	**42.30**	.03611	41.80	.02558	43.18	**.02484**	**45.02**	49.44
MMLU	.7685	32.68	**.4465**	**50.35**	**.06653**	**65.45**	.07061	65.25	**.03207**	66.01	.05497	**66.02**	68.43
IFEval	.9595	29.21	**.4266**	**57.49**	**.08640**	71.35	.08867	**73.38**	**.05441**	72.09	.05982	**72.64**	73.20
HumanEval	.2713	6.71	**.08496**	**37.80**	**.02035**	**60.98**	.02425	57.93	.01332	**65.24**	**.01251**	64.02	71.95
HellaSwag	.5664	38.06	**.2678**	**66.35**	**.04182**	**77.95**	.04654	77.60	**.02862**	**78.96**	.03995	78.46	80.41
Qwen3-8B (thinking disabled)													
	INT4				MXFP4				NVFP4				BF16
	WUSH (4.117)		TReQ (4.369)		WUSH (4.348)		TReQ (4.522)		WUSH (4.598)		TReQ (4.772)		
WikiText-2 PPL ↓	19.403		**14.358**		10.576		**10.487**		10.284		**10.259**		9.726
Dataset	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	KL ↓	Acc. ↑	Acc. ↑
MATH	.1289	51.94	**.08945**	**57.08**	**.01633**	**72.54**	.01768	72.04	**.01112**	73.74	.01190	**73.94**	75.40
MMLU	3.6732	36.59	**1.8721**	**58.70**	**.3314**	69.82	.5198	**70.72**	**.21863**	71.48	.28277	**71.79**	73.45
IFEval	.8549	55.45	**.5685**	**69.13**	**.1220**	**83.18**	.1371	80.04	.08881	79.48	**.08826**	**80.04**	80.59
HumanEval	.2813	29.88	**.1382**	**65.85**	.03486	**81.10**	**.03446**	79.88	.03102	**82.32**	**.02294**	79.88	84.76
HellaSwag	.2665	58.48	**.1692**	**64.41**	.03903	**73.99**	**.03777**	73.23	.02886	**75.26**	**.02636**	74.59	76.67

4.5 FUSED-KERNEL PERFORMANCE

On B200, paired native MXFP4 kernels measure the added execution cost of $b \times b$ output transformations (Figure 1). At $n = d_{\text{in}} = d_{\text{out}} = 4096$, adding the TF32 output transformation increases latency by approximately 1.5% and 5.6% for the 1-SM and 2-SM tile configurations, respectively, both below the 6.25% peak-throughput estimate in Section 3.1. At the tested workloads with at least 2,048 total tokens, keeping output transformations adds 0.2–1.5% (TF32) or 0.1–1.3% (FP16) to full-model prefill latency; native prefill is 1.24–$1.45\times$ faster than BF16. Small workloads can be slower than BF16. The no-$\boldsymbol{T}_{\text{out}}$ timing control removes transformations from the same checkpoint and computes a different function to test for the additional latency brought by the $\boldsymbol{T}_{\text{out}}$ transformation. The native backend also differs numerically from the quality-evaluation backend. For decoding with at most 16 rows, we measure a separate W4A8 operator (MXFP4 weights, MXFP8 activations). Output transformations add 0.23–0.42 μs per call to its GEMM kernel (3–9% of the operator), and the operator remains 1.02–$1.78\times$ faster than a BF16 matrix multiplication (Table 15). These results establish low incremental output-transformation cost for MXFP4 prefill and for the isolated decode operator at the stated workloads, not INT4 throughput or end-to-end decoding speed.

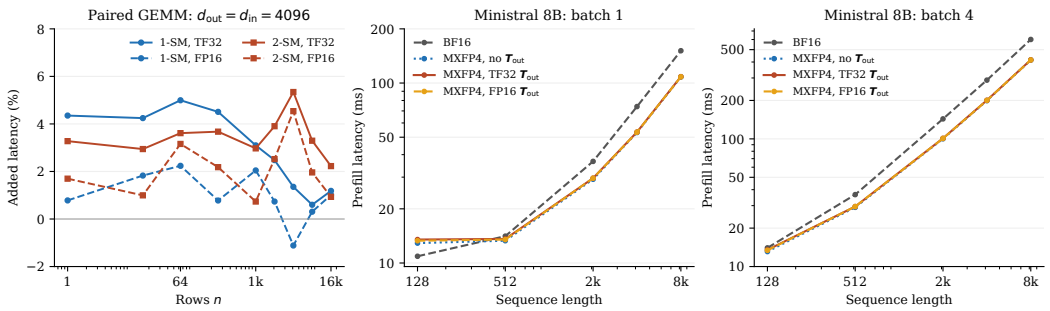

Figure 1: Native MXFP4 execution on B200. Left: incremental output-transformation cost of TF32 and FP16 output transformations relative to the same plain GEMM tile ($d_{\text{out}} = d_{\text{in}} = 4096$; CUDA Graph replay). Middle/right: full-model Ministral-3-8B prefill, batch sizes 1 and 4 (including input transformation and activation quantization; no graphs). The FP16 prefill variant retains TF32 for grouped Q/K/V projections. The no-$\boldsymbol{T}_{\text{out}}$ curve drops output transformations from the same checkpoint solely as a timing control; it computes a different function.

5 Conclusion

TReQ refits block-diagonal output transformations to match projection outputs, attention scores, and attention context from the original BF16 model, holding quantized weights and scales fixed during each refit. A fixed-checkpoint INT4 comparison isolates a KL benefit from refitting alone. Across three models, the full pipeline improves INT4 task fidelity and accuracy; FP4 block MSE gains are widespread but do not consistently transfer to full-model quality. These comparisons use one calibration draw per configuration and do not isolate individual construction components. Separately, B200 kernels demonstrate low incremental cost of output transformations in MXFP4 prefill and W4A8 decode operators, without establishing INT4 throughput or native-backend accuracy parity.

AI use statement

Generative AI tools assisted with retrieving and identifying related literature, code implementation and debugging, checking experimental and statistical results, interpreting results, and drafting and revising the manuscript. Verification included numerical checks and comparisons against saved evaluation outputs. The authors take responsibility for the final text, methods, experimental results, and claims.

Ethics statement

This work studies post-training quantization using existing pretrained language models and publicly available datasets. It involves no new data collection or experiments with human participants.

Reproducibility statement

Section 3 describes the algorithm, and Appendices A and C provide the optimization details and derivations. Appendix B documents model and dataset versions, calibration-data selection, transformation layouts, hyperparameters, evaluation metrics, and confidence-interval procedures, together with per-block results. Appendix B.8 specifies the hardware, software, and timing protocols for the kernel experiments.

REFERENCES

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*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b5b939436789f76f08b9d0da5e81af7c-Abstract-Conference.html.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M. De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/0df38cd13520747e1e64e5b123a78ef8-Abstract-Conference.html.

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

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/c3ba4962c05c49636d4c6206a97e9c8a-Abstract-Conference.html.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract.html.

Vage Egiazarian, Roberto L. 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 and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/1caf09c9f4e6b0150b06a07e77f2710c-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.

Han Guo, Philip Greengard, Eric Xing, and Yoon Kim. LQ-LoRA: Low-rank plus quantized matrix decomposition for efficient language model finetuning. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/d97a16cb83d74195b76e0bf1e85bf072-Abstract-Conference.html.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021b. URL https:

//datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

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

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/7183f4fc87598f6c6e947b96714acbd6-Abstract-Conference.html.

Junhan Kim, Ho-young Kim, Eulrang Cho, Chungman Lee, Joonyoung Kim, and Yongkweon Jeon. BoA: Attention-aware post-training quantization without backpropagation. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/kim25j.html.

Junhan Kim, Yeo Jeong Park, Seungwoo Son, Chungman Lee, Ho-young Kim, Joonyoung Kim, and Yongkweon Jeon. TurboBoA: Faster and exact attention-aware quantization without backpropagation. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/dd3bd4e35cdd224ed2153a996f41077f-Abstract-Conference.html.

Yuhang Li and Priyadarshini Panda. TesseraQ: Ultra low-bit LLM post-training quantization with block reconstruction. arXiv preprint arXiv:2410.19103, 2024. URL https://arxiv.org/abs/2410.19103.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. BRECQ: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=POWv6hDd9XH.

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*, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

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

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.

Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pp. 7197–7206, 2020. URL https://proceedings.mlr.press/v119/nagel20a.html.

NVIDIA. NVIDIA HGX Platform. URL https://www.nvidia.com/en-us/data-center/hgx/. Accessed September 26, 2026.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/c6483c8a68083af3383f91ee0dc6db95-Abstract-Conference.html.

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

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *International Conference on Machine Learning*, pp. 48630–48656, 2024. URL https://proceedings.mlr.press/v235/tseng24a.html.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, 2023. URL https://proceedings.mlr.press/v202/xiao23c.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. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. arXiv preprint arXiv:2311.07911, 2023. URL https://arxiv.org/abs/2311.07911.

A METHOD DETAILS

This appendix specifies the deployed operator and refitting systems from Section 3. Appendix C contains the initialization and its derivation; codec-specific construction settings appear in Appendix B.2.

A.1 TWO-SIDED LOW-PRECISION LINEAR OPERATOR

We use the notation of Section 3, with block size $b = 32$. The operators left_{32} and right_{32} apply the input and output transformations blockwise without materializing dense block-diagonal matrices.

Let $\mathcal{Q}_A$ denote activation quantization and let $\widehat{\boldsymbol{W}} = \mathcal{Q}_W(\boldsymbol{W}_{\mathrm{mid}})$ be the dequantized value of the frozen weight codes and scales. The activation quantizer likewise returns decoded numerical values in the equations below. Omitting bias, the TReQ model-quality operator uses the following precision path for all three codecs:

$$\widehat{\boldsymbol{A}} = \mathcal{Q}_A(\mathrm{left}_{32}\left(\mathrm{BF16}(\boldsymbol{A}), \mathrm{BF16}(\boldsymbol{T}_{\mathrm{in}})\right)), \tag{6}$$

$$\widehat{\boldsymbol{Y}} = \mathrm{BF16}\Big[\mathrm{right}_{32}\Big(\mathrm{TF32}(\widehat{\boldsymbol{A}}\widehat{\boldsymbol{W}}), \mathrm{TF32}(\boldsymbol{T}_{\mathrm{out}})\Big)\Big]. \tag{7}$$

Candidates are evaluated with the storage and compute precisions in Eqs. (6)– (7) before they are scored. Our reference follows the same surrounding precision,

$$\boldsymbol{Y}_{\mathrm{ref}} = \mathrm{BF16}(\mathrm{BF16}(\boldsymbol{A})\,\mathrm{BF16}(\boldsymbol{W}_0)). \tag{8}$$

Bias, when present, is restored at the same point and precision as in the inference implementation.

Before quantization, the input transformation, transformed weight, and output transformation obey

$$\boldsymbol{T}_{\mathrm{in}}\boldsymbol{W}_{\mathrm{mid}}\boldsymbol{T}_{\mathrm{out}} = \boldsymbol{W}_0. \tag{9}$$

This identity holds in exact arithmetic for invertible factors; Appendix C details the numerical safeguards and recomputation after storing the transformations.

The codec defines the decoded values returned by $\mathcal{Q}_A$ and $\mathcal{Q}_W$. NVFP4 applies its global scale in FP32 to each GEMM partial before the TF32 cast and output transformation. INT4 activation centering adds $\boldsymbol{\mu}\widehat{\boldsymbol{W}}$ to the centered quantized product before applying $\boldsymbol{T}_{\mathrm{out}}$, as specified in Appendix B.2. The activation-quantizer configuration is fixed during output refitting; its data-dependent scales are computed from each input.

Grouped deployment. Let $\boldsymbol{H}_{\mathrm{in}}$ be the normalized BF16 input second moment defined in Eq. (47). We divide the weight matrix into submatrices and give each its own transformations. Let $\{\Omega_p\}_{p=1}^{G_{\mathrm{out}}}$ partition output columns and $\{I_s\}_{s=1}^{G_{\mathrm{in}}}$ partition input rows. Here G_{out} is the number of output groups and G_{in} the number of input groups. Each submatrix defined by an input/output group pair is initialized and quantized separately:

$$\begin{gathered}\boldsymbol{W}_{s,p} = \boldsymbol{W}_0[I_s, \Omega_p], \qquad \boldsymbol{H}_s = \boldsymbol{H}_{\mathrm{in}}[I_s, I_s], \\ (\boldsymbol{T}_{\mathrm{in},s,p}, \widehat{\boldsymbol{W}}_{s,p}, \boldsymbol{T}_{\mathrm{out},s,p}) = \mathrm{QuantizeLinear}(\boldsymbol{H}_s, \boldsymbol{W}_{s,p}, \boldsymbol{A}[:, I_s]).\end{gathered} \tag{10}$$

For each output group p, assemble $\boldsymbol{T}_{\mathrm{in},p}$ from its $\boldsymbol{T}_{\mathrm{in},s,p}$ blocks across input groups. For each input group s, assemble $\boldsymbol{T}_{\mathrm{out},s}$ from its $\boldsymbol{T}_{\mathrm{out},s,p}$ blocks across output groups. Each submatrix uses its own moments and channel rescalings.

At inference, the low-precision GEMM returns an unsummed partial for every input group,

$$\boldsymbol{C}_{s,p} = \mathcal{Q}_A(\boldsymbol{A}\boldsymbol{T}_{\mathrm{in},p})[:, I_s]\widehat{\boldsymbol{W}}[I_s, \Omega_p], \qquad \widehat{\boldsymbol{Y}}[:, \Omega_p] = \sum_s \boldsymbol{C}_{s,p}\boldsymbol{T}_{\mathrm{out},s}|_{\Omega_p}, \tag{11}$$

Each partial is rounded to TF32 before its TF32 output transformation, held in an FP32 container. Transformed partials accumulate in FP32 in input-group order, and the sum is rounded once to BF16. Summing partial GEMM outputs before their distinct transforms, or BF16-rounding every transformed partial before summing, is a different numerical operator. The partials remain separate features when fitting the complete module output. Group boundaries align with the 32-channel transformations and the format's weight-scale groups; for attention projections we also align them with complete heads.

Proposition: diagonal and full output blocks. For one input group and $b = 32$ output channels J, fix $\boldsymbol{U}_J = (\boldsymbol{T}_{\text{in}}\widehat{\boldsymbol{W}})[:, J]$ and $\boldsymbol{W}_J = \boldsymbol{W}_0[:, J]$. In continuous weight reconstruction, with rounding, activation quantization, regularization, and invertibility constraints omitted,

$$L_{\text{full}} = \min_{\boldsymbol{B}} \|\boldsymbol{U}_J\boldsymbol{B} - \boldsymbol{W}_J\|_F^2 \le L_{\text{diag}} = \min_{\boldsymbol{D}\ \text{diagonal}} \|\boldsymbol{U}_J\boldsymbol{D} - \boldsymbol{W}_J\|_F^2.$$

Let $\boldsymbol{r}_k = \boldsymbol{w}_k - d_k^*\boldsymbol{u}_k$ be the optimal scalar-fit residual for column k, and let $\boldsymbol{P}_J$ project orthogonally onto $\text{col}(\boldsymbol{U}_J)$. Since the diagonal prediction lies in this space, orthogonal projection gives

$$L_{\text{diag}} - L_{\text{full}} = \sum_{k=1}^{b} \|\boldsymbol{P}_J\boldsymbol{r}_k\|_2^2.$$

Thus a nonzero projected residual gives strict improvement for this continuous comparison; the identity makes no claim about rounded, regularized fits.

A.2 Deriving transformed weights and refitting output transformations

After storing $\boldsymbol{T}_{\text{in}}$ in BF16 and rounding $\boldsymbol{T}_{\text{out}}$ to TF32, recompute $\boldsymbol{W}_{\text{mid}} = \boldsymbol{T}_{\text{in}}^{-1}\boldsymbol{W}_0\boldsymbol{T}_{\text{out}}^{-1}$. Recomputation requires relative full-product reconstruction error below 10^{-5}. This is an algebraic invariant before physical quantization, not a bound on deployed effective-weight error. Each input-side WUSH construction includes the normalized Hadamard factor defined in Appendix C.

Grouped least-squares notation. For an output block j of $b = 32$ channels, concatenate the fixed features from its input groups and stack their adjustable output blocks:

$$\boldsymbol{Z}_j = [\boldsymbol{C}_{1,j}\ \cdots\ \boldsymbol{C}_{G_{\text{in}},j}], \qquad \boldsymbol{\Theta}_j = [\boldsymbol{B}_{1,j}^\top\ \cdots\ \boldsymbol{B}_{G_{\text{in}},j}^\top]^\top. \tag{12}$$

The feature rows may represent projection outputs or attention contexts, as specified by each stage. For INT4, the projection features include the fixed centering offset $\boldsymbol{\mu}\widehat{\boldsymbol{W}}$, so its output correction follows the adjustable transformation. References to a fixed bias below mean the original module bias. Let $\boldsymbol{U}_j\boldsymbol{\Theta}_j$ be the effective weight on columns J_j; each input group's $\boldsymbol{T}_{\text{in}}\widehat{\boldsymbol{W}}$ occupies its corresponding input rows in $\boldsymbol{U}_j$. For fixed targets $\boldsymbol{Y}_j$, define

$$\boldsymbol{D}_j = \sum \boldsymbol{Z}_j^\top\boldsymbol{Z}_j, \quad \boldsymbol{R}_j = \sum \boldsymbol{Z}_j^\top\boldsymbol{Y}_j, \quad \boldsymbol{F}_j = \boldsymbol{U}_j^\top\boldsymbol{U}_j, \quad \boldsymbol{T}_j = \boldsymbol{U}_j^\top\boldsymbol{W}_0[:, J_j]. \tag{13}$$

The sum covers the fitting observations specified by the stage. Given positive energies E_D, E_W and coefficients γ_D, γ_W, the regularized normal system is

$$\left(\gamma_D\frac{\boldsymbol{D}_j}{E_D} + \gamma_W\frac{\boldsymbol{F}_j}{E_W} + \eta_j\boldsymbol{I}\right)\boldsymbol{\Theta}_j = \gamma_D\frac{\boldsymbol{R}_j}{E_D} + \gamma_W\frac{\boldsymbol{T}_j}{E_W}. \tag{14}$$

The off-diagonal blocks $\boldsymbol{C}_{s,j}^\top\boldsymbol{C}_{r,j}$ retain cross-group coupling: fitting every group independently to the complete target would count that target repeatedly. The stages below specify their different targets, energies, curvatures, feature precisions, and acceptance rules. Numerical damping η_j stabilizes the solve and is separate from the original-weight penalty.

Numerical details of the output-transformation fit. Let $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{W}_0$ denote the continuous fitting target, $\boldsymbol{Y}_{\text{ref}}$ the original BF16 layer output, and $E_Y = \|\boldsymbol{Y}_{\text{ref}}\|_F^2$ its energy. These distinguish the proposal's cross term from the rounded reference used for normalization and candidate scoring. Let subscript j denote the j-th b-column block of a matrix and $\boldsymbol{B}_j = (\boldsymbol{T}_{\text{in}}\widehat{\boldsymbol{W}})_j$. With $\gamma_r = (1-\lambda)/\kappa_r$ and $\gamma_w = \lambda/\kappa_w$, the continuous proposal for block j solves

$$\left(\gamma_r\frac{\boldsymbol{Z}_j^\top\boldsymbol{Z}_j}{E_Y} + \gamma_w\frac{\boldsymbol{B}_j^\top\boldsymbol{B}_j}{\|\boldsymbol{W}_0\|_F^2} + \mu_j\boldsymbol{I}\right)\boldsymbol{T}_{\text{out},j}^* = \gamma_r\frac{\boldsymbol{Z}_j^\top\boldsymbol{Y}_j}{E_Y} + \gamma_w\frac{\boldsymbol{B}_j^\top\boldsymbol{W}_{0,j}}{\|\boldsymbol{W}_0\|_F^2}. \tag{15}$$

The curvatures are the mean diagonal entries of the two normalized Gram matrices over all blocks, $\kappa_r = \|\boldsymbol{Z}\|_F^2/(d_{\text{out}}E_Y)$ and $\kappa_w = \|\boldsymbol{T}_{\text{in}}\widehat{\boldsymbol{W}}\|_F^2/(d_{\text{out}}\|\boldsymbol{W}_0\|_F^2)$, so the two terms contribute mean diagonals $1-\lambda$ and λ. They are frozen during one solve and its line search and recomputed when $\widehat{\boldsymbol{W}}$ or $\boldsymbol{T}_{\text{in}}$ changes. The ridge μ_j is 10^{-4} times the mean absolute diagonal of the matrix on the left, floored at FP64 machine epsilon; it is distinct from the effective-weight penalty. The Gram matrices

are formed from $\widehat{\boldsymbol{A}} = \mathcal{Q}_A(\boldsymbol{A}\boldsymbol{T}_{\text{in}})$ through $\boldsymbol{Z}_j^\top \boldsymbol{Z}_j = \widehat{\boldsymbol{W}}_j^\top \widehat{\boldsymbol{A}}^\top \widehat{\boldsymbol{A}} \widehat{\boldsymbol{W}}_j$ and $\boldsymbol{Z}_j^\top \boldsymbol{Y}_j = \widehat{\boldsymbol{W}}_j^\top \widehat{\boldsymbol{A}}^\top \boldsymbol{A}\boldsymbol{W}_{0,j}$, with $\widehat{\boldsymbol{A}}^\top \widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{A}}^\top \boldsymbol{A}\boldsymbol{W}_0$ accumulated in FP32 over streamed calibration inputs. The $b \times b$ systems are solved in FP64, and the solution is returned in FP32 and rounded to TF32. During separate construction of each input/output group pair, the target and weight energies cover that submatrix; subsequent coupled module refits use complete-projection energies as specified below. In the line search, candidates whose condition number exceeds 10^4 are rejected, since $\boldsymbol{W}_{\text{mid}}$ requires $\boldsymbol{T}_{\text{out}}^{-1}$. The procedure requires positive target energies and mean curvatures, invertible transformations, and solvable linear systems; if these conditions fail, the construction is stopped. The coefficient λ controls mean curvature contributions, not residual-loss percentages. At $\lambda = 1$, this local solve and its acceptance are weight-only; channel rescaling and RTN candidate selection still use the fitting inputs.

GPTQ with input and output transformations. We apply this pass once, to the RTN candidate selected in Section 3.2, and then refit $\boldsymbol{T}_{\text{out}}$ with the resulting codes and scales fixed. The GPTQ pass approximates a weight reconstruction. For fixed stored transforms its target is $\boldsymbol{P}_0 = \boldsymbol{T}_{\text{in}}^{-1}\boldsymbol{W}_0$ and its input moment is $\boldsymbol{H}_m = \boldsymbol{T}_{\text{in}}^\top \boldsymbol{H}_{\text{in}} \boldsymbol{T}_{\text{in}}$. To interpret the metric before numerical damping, let $\boldsymbol{E} = \widehat{\boldsymbol{W}} - \boldsymbol{W}_{\text{mid}}$ and $\boldsymbol{G} = \boldsymbol{T}_{\text{out}}\boldsymbol{T}_{\text{out}}^\top$. Using the normalized second moment from the BF16 calibration inputs $\boldsymbol{A}_c$ defined in Section 3.2,

$$\operatorname{tr}(\boldsymbol{E}^\top \boldsymbol{H}_m \boldsymbol{E}\boldsymbol{G}) = \frac{d_{\text{in}}}{\operatorname{tr}(\boldsymbol{A}_c^\top \boldsymbol{A}_c)} \|\boldsymbol{A}_c \boldsymbol{T}_{\text{in}} \boldsymbol{E} \boldsymbol{T}_{\text{out}}\|_F^2. \tag{16}$$

Thus the final GPTQ pass weights errors in the reconstructed linear map with activation quantization omitted. The subsequent output fit uses the quantized activations, as described in Section 3.2.

For each block of 32 output channels, set $\boldsymbol{M} = \boldsymbol{P}_{0,j}^\top$ and $\boldsymbol{B} = \boldsymbol{T}_{\text{out},j}^\top$. Output-coordinate error is weighted by $\boldsymbol{B}^\top \boldsymbol{B} = \boldsymbol{T}_{\text{out},j} \boldsymbol{T}_{\text{out},j}^\top$. The numerically ridged continuous start is $\boldsymbol{X} = (\boldsymbol{B}^\top \boldsymbol{B} + \mu \boldsymbol{I})^{-1} \boldsymbol{B}^\top \boldsymbol{M}$, with transform ridge fraction 10^{-4}. As damping tends to zero, it approaches the transformed weight derived from the pretrained weight.

Order output coordinates by descending diagonal of $\boldsymbol{B}^\top \boldsymbol{B}$. Let $\boldsymbol{D}$ be its ridged inverse in that order, and let $\boldsymbol{U}^\top \boldsymbol{U}$ equal the inverse of the symmetrized input moment with mean-diagonal damping fraction 10^{-2}. For each MXFP4 output row, freeze scales for groups of 32 input channels from its current values and round each contiguous group of 32 input channels together. Propagate its input-group error to unprocessed input coordinates by the inverse-Cholesky GPTQ update; there is no within-group feedback. After a full row is quantized, update each remaining output row s by $-(D_{sr}/D_{rr})(\boldsymbol{X}_r - \boldsymbol{q}_r)$. The inverse $\boldsymbol{D}$ is held fixed during the pass. This is a specified group-feedback heuristic, not exact discrete minimization. MXFP4 uses saturating E2M1 RTN codes (ties to even) and upward power-of-two rounding of $\max |\boldsymbol{x}_I|/6$ for 32-input-channel scales, with exponent limits $[-127, 127]$. An all-zero group can be represented by zero codes and a valid scale; it does not require an additional E8M0 encoding. Input groups are processed in order; scales are not refreshed during feedback. Codes remain frozen only in the subsequent output-refit step, not during this pass.

A.3 Q/K ATTENTION CALIBRATION

Optimizing Q, K, and V only as independent linear layers ignores how their errors interact inside attention. We therefore use the single-layer results as starting points with quantized weights fixed and refit only their block-diagonal output transformations. The input transformations, weight codes, native scales, and activation quantizer configuration remain fixed throughout this stage. We first describe attention without pre-RoPE Q/K RMSNorm.

Causal Q/K objective. Let H_Q and H_{KV} be the numbers of query and key/value heads, $r = H_Q/H_{KV}$, and $g(h) = \lfloor h/r \rfloor$ the KV head used by query head $h \in \{0, \ldots, H_Q - 1\}$. Let i index sequences, t query positions, u key positions, and d_{head} the number of channels per head. The vectors $\widehat{\boldsymbol{y}}^Q$ and $\widehat{\boldsymbol{y}}^K$ are the quantized projections' outputs after adding bias. Applying the position-dependent rotary matrices $\boldsymbol{R}_t$ and $\boldsymbol{R}_u$ gives the inference-time queries, keys, and scores below. A

star denotes the corresponding original BF16 quantity.

$$\widetilde{\boldsymbol{q}}_{iht} = \boldsymbol{R}_t \widehat{\boldsymbol{y}}^{Q}_{iht}, \qquad \widetilde{\boldsymbol{k}}_{igu} = \boldsymbol{R}_u \widehat{\boldsymbol{y}}^{K}_{igu},$$
$$\widetilde{S}_{ihtu} = \frac{\widetilde{\boldsymbol{q}}^{\top}_{iht} \widetilde{\boldsymbol{k}}_{i,g(h),u}}{\sqrt{d_{\text{head}}}}. \tag{17}$$

The target S^* uses the original BF16 Q and K projections, identical RoPE positions, the same grouped-query attention (GQA) head mapping $g(h)$, and the same causal mask. We minimize the normalized error over causally allowed attention scores

$$\mathcal{L}_{\text{score}} = \frac{\sum_{i,h,t} \sum_{u \le t} (\widetilde{S}_{ihtu} - S^*_{ihtu})^2}{\sum_{i,h,t} \sum_{u \le t} (S^*_{ihtu})^2}. \tag{18}$$

For side $a \in \{Q, K\}$, let $\boldsymbol{\theta}_a$ collect its output-transformation parameters and let p_a be their count. Let $\mathcal{J}_a$ map an increment in these parameters to the unrounded score increment with the other side fixed; its explicit form is given below. The adjoint $\mathcal{J}_a^*$ is its transpose under the usual sum-of-products inner product. Define E_S as the denominator in Eq. (18). For each side, $\mathcal{R}_{\text{eff},a}$ is the squared error of its effective weight relative to its pretrained weight, divided by that pretrained weight's squared Frobenius norm. The fitting objective combines Eq. (18) with effective-weight errors against the two original projection weights, again after normalizing both mean curvatures:

$$\mathcal{L}_{QK} = \frac{1 - \lambda_{QK}}{\kappa_{\text{score}}} \mathcal{L}_{\text{score}} + \frac{\lambda_{QK}}{\kappa_{\text{eff}}} \left(\mathcal{R}_{\text{eff},Q} + \mathcal{R}_{\text{eff},K} \right). \tag{19}$$

Each stage computes its own mean curvatures at entry and freezes them through its sweeps and line searches. The score curvature is the parameter-count-weighted mean of $\operatorname{tr}(\mathcal{J}_a^* \mathcal{J}_a)/(p_a E_S)$ for $a \in \{Q, K\}$, with the opposite side fixed. The weight curvature similarly averages the normalized effective-weight Gram traces, using each side's complete original-weight energy and parameter count. These traces set the coefficients; all off-diagonal couplings remain in the solves. Both the projected and direct stages use causal-score curvature, so the projected stage does not rebalance using its desired-output Gram matrices. Its desired-target energies, however, change at each side update. Alternating updates hold the other side fixed. Table 5 gives the coefficients for the single-layer, Q/K, V, and downstream stages.

Initialization by fitting desired Q/K outputs. The projected Q/K warm start first solves for desired Q/K output vectors, then fits the transformations to those vectors. These intermediate fitting targets generally differ from the BF16 projection outputs. With K fixed, the desired rotated query at position t solves

$$\boldsymbol{q}'_{iht} = \arg\min_{\boldsymbol{x}} \sum_{u \le t} \left(\boldsymbol{x}^{\top} \widetilde{\boldsymbol{k}}_{i,g(h),u} - {\boldsymbol{q}^*_{iht}}^{\top} \boldsymbol{k}^*_{i,g(h),u} \right)^2 + \mu \|\boldsymbol{x}\|_2^2, \tag{20}$$

where μ is a small diagonal regularizer used to stabilize the solve. The desired key output is the reverse-time analogue and sums contributions from every query head mapped to the KV head.

For side $a \in \{Q, K\}$, let $\boldsymbol{Y}_a$ stack these desired outputs after inverse RoPE and subtraction of the fixed stored bias, over all fitting sequences, positions, and that side's output channels. Set $E_a^{\text{raw}} = \|\boldsymbol{Y}_a\|_F^2$ and $E_{W_a} = \|\boldsymbol{W}_{0,a}\|_F^2$ for its complete original weight. For output block j with channel set J_j, let $\boldsymbol{Z}_{a,j}$ concatenate the TF32-rounded frozen partials from all input groups, and let $\boldsymbol{\Theta}_{a,j}$ stack the adjustable output-transformation blocks. Use $\boldsymbol{U}_{a,j}$ and the grouped statistics from Eq. (13) for this side. With $\gamma_r = (1 - \lambda_{QK})/\kappa_{\text{score}}$ and $\gamma_w = \lambda_{QK}/\kappa_{\text{eff}}$ fixed at projected-stage entry, the continuous fitting objective is

$$\sum_j \left[\frac{\gamma_r}{E_a^{\text{raw}}} \|\boldsymbol{Z}_{a,j} \boldsymbol{\Theta}_{a,j} - \boldsymbol{Y}_a[:, J_j]\|_F^2 + \frac{\gamma_w}{E_{W_a}} \|\boldsymbol{U}_{a,j} \boldsymbol{\Theta}_{a,j} - \boldsymbol{W}_{0,a}[:, J_j]\|_F^2 \right]. \tag{21}$$

The proposal uses Eq. (14) with $E_D = E_a^{\text{raw}}$, $E_W = E_{W_a}$, $\gamma_D = \gamma_r$, and $\gamma_W = \gamma_w$. Q and K have separate target energies, recomputed at every side update and sweep. K's energy counts its actual KV-head channels once, although its desired-key solve aggregates all mapped query heads.

Each Q proposal is line-searched in stored TF32 on the regularized deployed fit-score objective; K is then proposed using the accepted Q and similarly line-searched. The complete projected-stage pair is accepted or rolled back on internal-validation score error. After that acceptance or rollback, the direct stage initializes its own curvature-balancing scalars.

Direct RoPE-aware ALS refinement. After the projected stage, we alternate conditional score solves: Q with K fixed, then K with the updated Q fixed. For dot-product attention without Q/K RMSNorm, the unrounded score surrogate is linear in the current side, but the actual update uses the affine local surrogate defined below to retain the current deployment residual. For a Q-side update and input group s, let $\boldsymbol{C}_s^Q$ be the corresponding TF32-rounded frozen-code partial product and $\Delta\boldsymbol{B}_s$ the block-diagonal change to that group's Q output transformation. Its contribution to a score change is

$$\mathcal{J}_s^Q(\Delta\boldsymbol{B}_s)_{ihtu} = \mathbb{1}[u \le t]\,\frac{\left\langle \boldsymbol{R}_t(\boldsymbol{C}_s^Q\Delta\boldsymbol{B}_s)_{iht}, \widetilde{\boldsymbol{k}}_{i,g(h),u}\right\rangle}{\sqrt{d_{\text{head}}}}, \qquad \mathcal{J}^Q = \sum_s \mathcal{J}_s^Q. \tag{22}$$

Let $\boldsymbol{\theta}_0$ vectorize the current side's stored output-transformation blocks, and let $\Delta\boldsymbol{\theta}$ vectorize their proposed changes. Writing $\mathcal{J}$ for the current side's score-change map, define

$$\boldsymbol{S}_{\text{sur}}(\Delta\boldsymbol{\theta}) = \widetilde{\boldsymbol{S}}(\boldsymbol{\theta}_0) + \mathcal{J}\Delta\boldsymbol{\theta}, \qquad \boldsymbol{e} = \widetilde{\boldsymbol{S}}(\boldsymbol{\theta}_0) - \boldsymbol{S}^*. \tag{23}$$

The offset is the current deployed score, including rounding; the Jacobian uses these frozen partials. The solve is exact for this conditional quadratic surrogate (up to numerical damping), not for the discontinuous deployed map or necessarily for its globally unrounded counterpart. With block size $b = 32$, head dimension d_{head} and G_{in} input groups, each side/head has $p = G_{\text{in}}(d_{\text{head}}/b)b^2 = G_{\text{in}}bd_{\text{head}}$ scalar transform variables. The dense direct Gram is $p \times p$, with all input-group and within-head block couplings. For $G_{\text{in}} = 2, d_{\text{head}} = 128$, $p = 8192$: its FP32 Gram alone occupies 256 MiB, excluding factorization workspace. This is larger than fitting an individual projection's output transformation. For the current side, write its unnormalized effective-weight penalty as $\boldsymbol{\theta}^\top\boldsymbol{G}_{\text{eff}}\boldsymbol{\theta} - 2\boldsymbol{\theta}^\top\boldsymbol{c}_{\text{eff}} + \text{const}$. This defines the Gram matrix $\boldsymbol{G}_{\text{eff}}$ and target cross-product $\boldsymbol{c}_{\text{eff}}$ in the vectorized parameters. Let E_W be the squared norm of that side's complete pretrained weight, $\gamma_r = (1-\lambda_{QK})/\kappa_{\text{score}}$, and $\gamma_w = \lambda_{QK}/\kappa_{\text{eff}}$. Using the score energy E_S defined above, the update solves

$$\left[\gamma_r\frac{\mathcal{J}^*\mathcal{J}}{E_S} + \gamma_w\frac{\boldsymbol{G}_{\text{eff}}}{E_W} + \mu\boldsymbol{I}\right]\Delta\boldsymbol{\theta} = -\gamma_r\frac{\mathcal{J}^*\boldsymbol{e}}{E_S} - \gamma_w\frac{\boldsymbol{G}_{\text{eff}}\boldsymbol{\theta}_0 - \boldsymbol{c}_{\text{eff}}}{E_W}. \tag{24}$$

The second right-hand-side term penalizes deviation from the original weight, not from the current transformation. Query heads have independent normal systems, while each KV-head system sums all query heads assigned to it. All input groups and transform blocks within a head remain coupled.

We accumulate $\mathcal{J}^*\mathcal{J}$ and $\mathcal{J}^*\boldsymbol{e}$ from the following explicit contractions. For one Q head, let $F_t \in \mathbb{R}^{d_{\text{head}}\times p}$ map vectorized input-group/block transform increments to the raw query increment at t. Define $K_t = K_{t-1} + \widetilde{k}_t\widetilde{k}_t^\top$, $K_{-1} = 0$. Its Gram contribution is $F_t^\top R_t^\top K_t R_t F_t/d_{\text{head}}$; its residual contribution is $F_t^\top R_t^\top \sum_{u\le t} e_{tu}\widetilde{k}_u/\sqrt{d_{\text{head}}}$. Sum over positions and sequences. For K, use suffix $Q_u = Q_{u+1} + \sum_{h:g(h)=g}\widetilde{q}_{hu}\widetilde{q}_{hu}^\top$ and residual sums over $t \ge u$ and mapped query heads. These formulas retain cross-group and cross-block normal entries. Prefix/suffix covariances avoid enumerating every score Jacobian, but each head still needs $O(p^2)$ Gram storage and $O(p^3)$ Cholesky work. We accumulate these contractions using streamed causal prefix statistics for Q and suffix statistics for K, avoiding a retained batch-by-head score tensor and a model-wide Hessian; an individual sequence/head residual may still be materialized. Each side has a stored-TF32 line search on the regularized fit objective before updating the other side. Let $\boldsymbol{T}_{\text{out},a}^{\text{entry}}$, $a \in \{Q, K\}$, denote the pair saved at direct-stage entry, including any accepted projected updates. Let $\widehat{\boldsymbol{T}}_{\text{out},a}$ denote the pair after all direct fitting sweeps and per-side line searches. We jointly interpolate between these endpoints using the same step size for Q and K:

$$\boldsymbol{T}_{\text{out},a}(\alpha) = \text{TF32}\left[\boldsymbol{T}_{\text{out},a}^{\text{entry}} + \alpha(\widehat{\boldsymbol{T}}_{\text{out},a} - \boldsymbol{T}_{\text{out},a}^{\text{entry}})\right], \quad \alpha \in \{1, \tfrac{1}{2}, \tfrac{1}{4}, \tfrac{1}{8}\}. \tag{25}$$

The zero-step candidate uses the saved entry pair without re-rounding. Among candidates satisfying the condition-number checks, we select a strict improvement over the entry pair's causal-score relative MSE on complete internal-validation sequences; otherwise, we return the entry pair unchanged. The weight penalty affects the proposal but is not part of this final selection criterion. The conditional-linear derivation above applies to attention without pre-RoPE Q/K RMSNorm; recurrent attention is outside this setup.

Numerical settings for Q/K and V solves. For the desired-vector solve, adjacent causal systems differ by rank-one updates, so their inverses can be updated with the Woodbury identity and periodically refactorized. For a query position the desired-output system has dimension d_{head}, prefix Gram $K_t = K_{t-1} + k_t k_t^\top$, and cross matrix accumulating deployed-key/teacher-key outer products. That cross matrix times the teacher query gives the right-hand side. Keys use suffix statistics with rank $r = H_Q/H_{KV}$ updates, summing mapped query heads. Initialize the inverse from a ridge of 10^{-6} times full-sequence squared feature norm divided by d_{head}, floored at 10^{-12}. For update columns U, apply $P \leftarrow P - PU(I + U^\top PU)^{-1}U^\top P$. Refactor every 32 visible positions; failed factorizations increase jitter tenfold for at most six retries. The subsequent grouped output-transformation fit has Gram order $32G_{\mathrm{in}}$ for each block of 32 output channels and 32 right-hand sides, distinct from the d_{head}-dimensional desired-output solve. The desired-output ridge is zero-centered numerical stabilization, not an anchor to the current outputs.

For direct Q/K, heads are processed serially, assembling and Cholesky-solving in FP32. Damping is the greater of 10^{-6} times nonnegative mean diagonal and 10^{-12}, increased tenfold for at most eight factorization attempts.

For both the projected Q/K transformation fit and the V fit below, let $\boldsymbol{A}_j$ be the combined normal matrix in Eq. (14) before damping, of order $m = 32G_{\mathrm{in}}$. Use $\eta_j = 10^{-6}\max(|\operatorname{tr}(\boldsymbol{A}_j)/m|, \epsilon_{64})$, where ϵ_{64} is FP64 machine epsilon. These systems are solved in FP64 and return FP32 proposals. The ridge is zero-centered and does not shift the right-hand side toward the current transformation.

Q/K head normalization in Qwen3. Qwen3 applies a learned RMSNorm to each Q/K head before RoPE. For a head vector $\boldsymbol{z} \in \mathbb{R}^d$, write $\mathcal{N}(\boldsymbol{z}) = \boldsymbol{\gamma} \odot \boldsymbol{z}/r$ with $r = (\|\boldsymbol{z}\|_2^2/d + \epsilon)^{1/2}$ and fixed learned gain $\boldsymbol{\gamma}$. Its differential is

$$D\mathcal{N}_{\boldsymbol{z}}[\boldsymbol{v}] = \boldsymbol{\gamma} \odot \left(\frac{\boldsymbol{v}}{r} - \frac{\boldsymbol{z}(\boldsymbol{z}^\top \boldsymbol{v})}{dr^3}\right). \tag{26}$$

After single-layer construction, we first refit Q and K against their original BF16 projection outputs after this normalization. Each head's update applies a common block-diagonal right increment to its existing grouped output transformations. The objective combines relative normalized-output reconstruction error with the effective-weight penalty. We form Gauss–Newton proposals using Eq. (26) and solve them with matrix-free preconditioned conjugate gradients. We use two Gauss–Newton steps, with PCG tolerance 10^{-6} and a cap of 8,000 iterations. A six-step halving line search evaluates the rounded nonlinear computation and accepts only a decrease in the regularized fitting objective, subject to a transformation condition number at most 10^4.

We disable the projected Q/K warm start for this model. The subsequent alternating score refinement composes each projection-output increment with $D\mathcal{N}$ before RoPE in Eq. (22). This gives a local Gauss–Newton score system; each fitting line search evaluates the actual normalized, rounded attention scores. For the final joint Q/K interpolation, validation uses attention-output relative MSE: candidate Q/K probabilities multiply the reference BF16 values and pass through the reference BF16 O projection, with the original BF16 attention output as target. V and O stay fixed during this selection. We retain the entry pair unless a candidate strictly improves this validation error.

A.4 Attention-aware V refit

Hold Q/K, V's input transformations, codes, scales, and bias fixed. Let $\widetilde{\boldsymbol{P}}_{ih}$ and $\boldsymbol{P}^*_{ih}$ be the causal attention probabilities from the current and reference Q/K, respectively, and let $\boldsymbol{V}^*_{ig}$ be the original V projection on the same inputs. These probabilities and reference values are stored in BF16. The continuous proposal uses the reference product

$$\boldsymbol{C}^*_{ih} = \mathrm{FP32}(\boldsymbol{P}^*_{ih})\,\mathrm{FP32}(\boldsymbol{V}^*_{i,g(h)}), \qquad E_C = \sum_{i,h} \|\boldsymbol{C}^*_{ih}\|_F^2, \tag{27}$$

without a final BF16 context cast. Here and below i ranges over fitting sequences, and the norm includes all query positions and head channels. The one denominator E_C covers all query heads, including each head sharing a KV head, before any bias subtraction.

Here j enumerates V's 32-channel output blocks across all KV heads. Let g_j be its KV head and $J_j \subset \{0, \ldots, d_{\mathrm{head}} - 1\}$ its 32 head-local coordinates. The corresponding full-projection columns

are $\bar{J}_j = \{g_j d_{\text{head}} + c : c \in J_j\}$. For input group s, let $\boldsymbol{M}_{s,ij}$ be its TF32-rounded frozen V partial. Concatenate the features $\mathrm{FP32}(\widetilde{\boldsymbol{P}}_{ih})\boldsymbol{M}_{s,ij}$ over input groups to form $\boldsymbol{Z}_{ih,j}$, for $g(h) = g_j$. Let $\boldsymbol{\Theta}_j$ stack the corresponding adjustable output blocks, and let $\boldsymbol{b}_{V,j}$ be the fixed stored V bias on $\bar{J}_j$. With $\mathbf{1}$ a column of ones over value positions, set

$$\boldsymbol{Y}_{ih,j} = \boldsymbol{C}^*_{ih}[:, J_j] - \mathrm{FP32}(\widetilde{\boldsymbol{P}}_{ih})(\mathbf{1}\boldsymbol{b}_{V,j}). \tag{28}$$

The continuous context prediction is $\boldsymbol{Z}_{ih,j}\boldsymbol{\Theta}_j$ plus this fixed attention-weighted bias. It does not add the current context's deployment-rounding residual. Restoring bias in this form also accounts for BF16 probability rows whose sums differ from one.

Use Eq. (13) with column set $\bar{J}_j$ for the effective-weight map, and sum the data statistics $\boldsymbol{D}_j, \boldsymbol{R}_j$ over sequences i and mapped query heads $h : g(h) = g_j$. Set $E_W = \|\boldsymbol{W}_{0,V}\|_F^2$. The weight statistics $\boldsymbol{F}_j, \boldsymbol{T}_j$ are counted once per V output block, while the data statistics include every mapped query head. Let n_b be the number of output blocks and $m = 32G_{\text{in}}$ the order of each normal matrix. For regularization coefficient λ_V, define projection-wide mean curvatures and weights

$$\kappa_C = \frac{\sum_j \operatorname{tr} \boldsymbol{D}_j}{n_b m E_C}, \quad \kappa_W = \frac{\sum_j \operatorname{tr} \boldsymbol{F}_j}{n_b m E_W}, \quad \gamma_C = \frac{1-\lambda_V}{\kappa_C}, \quad \gamma_W = \frac{\lambda_V}{\kappa_W}. \tag{29}$$

All blocks have the same number of right-hand sides, so these averages also equal averages over the scalar transform parameters. The continuous objective is

$$\frac{\gamma_C}{E_C} \sum_{i,h,j:g_j=g(h)} \|\boldsymbol{Z}_{ih,j}\boldsymbol{\Theta}_j - \boldsymbol{Y}_{ih,j}\|_F^2 + \frac{\gamma_W}{E_W} \sum_j \|\boldsymbol{U}_j\boldsymbol{\Theta}_j - \boldsymbol{W}_{0,V}[:, \bar{J}_j]\|_F^2. \tag{30}$$

These energies and coefficients are fixed throughout the V refit. The proposal follows Eq. (14) with $E_D = E_C$ and $\gamma_D = \gamma_C$; its numerical settings are listed above with those of the projected Q/K transformation fit.

Candidate evaluation uses stored-TF32 steps toward this proposal. It computes BF16 V outputs, then rounds both candidate and reference contexts to BF16; the reference is now $\boldsymbol{C}^{*,\text{dep}} = \mathrm{BF16}(\boldsymbol{C}^*)$. Both contexts pass through the original BF16 O projection, producing BF16 outputs. O does not enter the V normal equations. Recorded context relative MSE uses the energy of $\boldsymbol{C}^{*,\text{dep}}$ on its evaluation split, while module relative MSE uses the energy of the reference output after O. Among feasible candidates, selection requires a strict decrease in internal-validation module relative MSE; the weight penalty affects the proposal but is not added to this acceptance metric. We finish quantizing and refitting Q, K, and V before quantizing O.

A.5 Sequential model calibration

Before any projection is replaced, we replay the cached BF16 block inputs through the original block and record the attention output projection and MLP down projection outputs. Denote either fixed reference by $\boldsymbol{T}^*$; it includes the projection's bias and precedes residual addition. These references remain fixed as we replay the partially quantized block to obtain current fitting inputs.

Fitting inputs must reflect the errors already introduced upstream. We therefore traverse each transformer block in execution order. Before each projection (or the combined Q/K/V stage), we replay the current partially quantized block on reconstruction sequences and collect its fitting inputs. The construction statistic $\boldsymbol{H}_{\text{in}}$ retains its precomputed BF16 value; the fitting Gram matrices and cross-products are formed from the current fitting inputs and features. We then quantize and refit the projection, validate the update, and install the replacement. For a Llama-style block the order is

$$\text{joint QKV} \;\to\; O \;\to\; \text{gate} \;\to\; \text{up} \;\to\; \text{down}. \tag{31}$$

Attention calibration uses complete contiguous sequences to preserve RoPE, causality, and Q/K/V pairing.

For O and down, keep the input transformations, weight codes, scales, and bias fixed and fit the output blocks using Eq. (14). Here $\boldsymbol{Z}_j$ concatenates partial GEMM outputs from the current transformed, quantized and decoded activations and frozen decoded weights. These products stay in FP32, with no TF32 rounding before Gram and cross-product accumulation; WUSH uses the same convention with its own activation quantizer and weight decoder. Set

$$\boldsymbol{Y}^* = \boldsymbol{T}^* - \mathbf{1}\boldsymbol{b}, \qquad E_D = \sum \|\boldsymbol{Y}^*\|_F^2, \qquad E_W = \|\boldsymbol{W}_0\|_F^2, \tag{32}$$

where $\boldsymbol{b}$ is the fixed stored bias and $\boldsymbol{Y}_j = \boldsymbol{Y}^*[:, J_j]$. The data statistics sum over fitting sequences. Both energies cover the complete projection; block-specific statistics use only J_j. For n_b output blocks and normal-matrix order $m = 32G_{\text{in}}$, use $\gamma_D = (1-\lambda)/\kappa_D$ and $\gamma_W = \lambda/\kappa_W$, where

$$\kappa_D = \frac{\sum_j \operatorname{tr} \boldsymbol{D}_j}{n_b m E_D}, \qquad \kappa_W = \frac{\sum_j \operatorname{tr} \boldsymbol{F}_j}{n_b m E_W}.$$

A small diagonal regularizer stabilizes the solve. Fitting and internal-validation sequences are disjoint. Validation restores the rounded output-transformation path, bias, and BF16 projection-output rounding, and normalizes the squared error by $\sum \|\boldsymbol{T}^*\|_F^2$ on that split. Accept only a strict decrease. For both O and down, the target is the original BF16 projection output before residual addition; no residual term enters fitting or acceptance. Gate and up retain their single-layer fits and do not receive this fixed-reference downstream refit.

Model blocks may be calibrated in parallel by supplying each worker with the cached input of the corresponding original BF16 block. Within a block, however, the order in Eq. (31) is preserved.

B Experimental Details

This appendix describes the matched Tülu-calibrated comparisons, their evaluation protocols, and the controlled experiments.

B.1 Configuration and calibration

The three text towers use the configurations in Table 3. The quantized projections are Q/K/V/O/gate/up/down; embeddings, normalization, and the LM head retain higher precision. Ministral uses `mistralai/Ministral-3-8B-Instruct-2512-BF16`, revision

Table 3: Models used in the matched comparisons.

Model	Transformer blocks
Ministral-3-8B-Instruct-2512-BF16	34
Llama-3.1-8B-Instruct	32
Qwen3-8B	36

`f6fae9795746f63c9be8344932f01275f3c63734`, with 238 quantized projections. Llama uses revision `0e9e39f249a16976918f6564b8830bc894c89659`, with 224 quantized projections. Qwen3 uses revision `b968826d9c46dd6066d109eabc6255188de91218` and includes query/key RMS normalization. Model-quality evaluation uses numerical quantize/dequantize on GH200; native MXFP4 performance is measured separately on B200 (Appendix B.8). Input transformations use BF16 storage; output transformations use FP32 containers with TF32-rounded values. Execution uses TF32 Tensor Cores with FP32-range intermediates. The WUSH comparison uses matched one-sided adaptations for INT4 and MXFP4. Both adaptations use GPTQ rather than rounding alone. The MXFP4 adaptation uses 32-wide activation, weight, and Hadamard groups; INT4 uses per-token activation scales and per-output-channel weight scales. The official implementation reference is commit `8b1589f71db3da73f1775e84c2359446b585d548`.

All model comparisons use Tülu chat calibration. The reported model-level comparisons use one calibration draw per configuration. Codec-specific balancing means that INT4 versus MXFP4 scores are not a datatype-only ablation. Within each model, the INT4 and MXFP4 comparisons share the input-second-moment capture. Ministral input moments use 51,455 complete conversations (67.2 million tokens), with 96 additional conversations for fitting, 32 for internal validation, and 32 held out. Qwen3 input moments use 51,980 complete conversations (42.0 million tokens). Llama input moments use 51,455 complete conversations (42.0 million tokens). The chat-fidelity panels score 32 fitting conversations and 32 held-out conversations, excluding internal validation. For Ministral, the 32 fitting conversations are a subset of its 96-conversation fitting pool. We use the training split of `allenai/tulu-3-sft-mixture`, revision `b14afda60f1bbebe55d5d2fa1e4df5042f97f8be`. Ministral conversations are selected from a seed-42 full-index shuffle, excluding empty messages, normalized-text duplicates,

and matches to normalized HumanEval prompts, function signatures, or long reference solutions. Moment capture rounds each worker's token budget up to a complete conversation. The fitting, validation, and held-out pools are then drawn from unused conversations with 1,024–2,048 Ministral-template tokens, without truncation or padding. Llama retokenizes the same raw conversation IDs with its native template. Qwen3 excludes moment-capture conversations exceeding its 40,960-token context limit and replaces their source-token allocation using a seed-43 sample of unused complete conversations; its fitting and evaluation pools retain the shared raw IDs.

Table 4: MXFP4 and NVFP4 layouts, fixed before and after refitting. G_{out} counts output groups and G_{in} counts input groups. Each input/output group pair has its own transformations.

Projection	G_{out}	G_{in}
Q, K, V	2	2
Attention output	1	1
MLP gate, up, down	1	1

Groups are equal, contiguous, and aligned with 32-wide transforms and complete attention heads. INT4 uses $G_{\text{out}} = 1$, $G_{\text{in}} = 2$ for all seven projections in each block of all three models. Each input/output group pair is quantized independently; subsequent attention refits optimize their output transformations together. Across all 714 INT4 projections in 102 blocks, each projection stores two equal input tiles spanning the complete output dimension, with 32-wide BF16 input transformations and FP32 output-transformation containers. Packed codes use `uint8` storage; channel scales, centering means, and bias buffers use FP32.

Table 5: Fixed curvature-balanced regularization coefficients in $[0, 1]$. Numerical damping is separate. Coefficients are fixed before full-model evaluation.

Stage	λ
Single-layer output refit	0.1
Projected Q/K warm start	0.003
Direct Q/K ALS	0.3
V context refit	0.4
O/down fit to BF16 outputs	0.2

The direct stage uses one Q-then-K sweep; projected warm start has a two-sweep cap and validation rollback. Score batching processes one sequence at a time. Across codecs, RTN candidates are ranked by fitting-set reconstruction error with inference-time rounding; only the selected candidate receives one GPTQ pass and a final output refit. For Qwen3, the single-linear coefficient is 0.1; the RMS-normalized projection and joint-QK coefficients are 0.3, V uses 0.4, and O/down uses 0.2. Projected Q/K warm start is disabled. Attention and downstream teacher normal systems use numerical damping 10^{-6}, compared with 10^{-4} for single layers. These damping terms stabilize the solvers; weight penalties have separate coefficients. The construction uses four input/weight updates for INT4, two for MXFP4, and five for NVFP4.

Block construction. Input second moments are computed from BF16 activations and held fixed during quantization; blocks start from cached BF16 inputs, and their internal activations are recomputed after preceding quantization and refit steps. INT4 uses signed codes in $[-7, 7]$, dynamic per-token activation scales, and per-output-channel weight scales. The WUSH comparator is our adaptation of the published one-sided construction to this codec, with one GPTQ pass; it is not an INT4 result reported by the WUSH paper.

B.2 Codec-specific rescaling and centering

The following rescaling rules are codec-specific construction settings. For MXFP4, channel rescaling precedes quantization. We use positive diagonal matrices $\boldsymbol{D}_{\text{in}}$ and $\boldsymbol{D}_{\text{out}}$ to redistribute channel magnitudes, compensating adjacent factors to preserve their product:

$$\boldsymbol{T}'_{\text{in}} = \boldsymbol{T}_{\text{in}}\boldsymbol{D}_{\text{in}}^{-1}, \qquad \boldsymbol{W}'_{\text{mid}} = \boldsymbol{D}_{\text{in}}\boldsymbol{W}_{\text{mid}}\boldsymbol{D}_{\text{out}}^{-1}, \qquad \boldsymbol{T}'_{\text{out}} = \boldsymbol{D}_{\text{out}}\boldsymbol{T}_{\text{out}}. \tag{33}$$

This rescaling preserves Eq. (9) before rounding. The diagonal entries are continuous in the evaluated configuration.

Quantization formats. INT4 uses signed codes in $[-7, 7]$, per-token activation scales and per-output-channel weight scales. MXFP4 uses E2M1 codes and one E8M0 power-of-two scale per 32 consecutive input channels. NVFP4 uses E2M1 codes, E4M3 scales per 16 input channels, and FP32 global scales. Its input/output transformations remain 32-wide. The NVFP4 construction prescreens global-scale candidates using diagonal-Hessian-weighted weight reconstruction error with RTN, preferring candidates without scale overflow, before the terminal GPTQ pass.

MXFP4 rescaling. For MXFP4, we equalize root-mean-square (RMS) magnitudes within 32-channel groups. Let

$$\overline{\boldsymbol{A}} = \mathrm{BF16}(\boldsymbol{A})\,\mathrm{BF16}(\boldsymbol{T}_{\mathrm{in}}), \qquad m_k = \sqrt{\frac{1}{n}\sum_s \overline{A}_{sk}^2}. \tag{34}$$

Within each scale group $\mathcal{B}$ of 32 input channels, let $\mathcal{B}_+ = \{k \in \mathcal{B} : m_k > 0\}$. For nonempty $\mathcal{B}_+$, we center the nonzero channel RMS values around their geometric mean,

$$g_{\mathrm{in},\mathcal{B}} = \exp\left(\frac{1}{|\mathcal{B}_+|}\sum_{k\in\mathcal{B}_+}\log m_k\right), \qquad \delta_{\mathrm{in},k} = \frac{m_k}{g_{\mathrm{in},\mathcal{B}}}. \tag{35}$$

Channels with zero magnitude retain scale one and are excluded from the geometric mean. For the remaining channels, $\log_2 \delta$ is clipped to $[-24, 24]$. The output-channel rescaling is computed from $\boldsymbol{V} = \boldsymbol{D}_{\mathrm{in}}\boldsymbol{W}_{\mathrm{mid}}$. If $r_c = (d_{\mathrm{in}}^{-1}\sum_k V_{kc}^2)^{1/2}$, its columns are centered in the same way within contiguous 32-channel output groups to obtain $\delta_{\mathrm{out},c}$. This equalization reduces magnitude differences within each group.

After RMS equalization, a further numerical-range safeguard rescales channels whose maximum absolute value exceeds 400. For a channel maximum m, the additional divisor is $s(m) = \min(2^{24}, \max(1, m/400))$. We first apply it to transformed activations, dividing columns of $\boldsymbol{T}_{\mathrm{in}}$ and multiplying the matching rows of the transformed weight. Because $\boldsymbol{T}_{\mathrm{in}}$ is stored in BF16, we recompute activation maxima after rounding and apply up to three additional corrections, capping the cumulative divisor for this safeguard at 2^{24}. We then apply the same rule to columns of the compensated transformed weight, multiplying matching rows of $\boldsymbol{T}_{\mathrm{out}}$. All rescalings preserve the result of the matrix multiplication before rounding. Both diagonals are folded into the factors before constructing codes. After rounding the transforms we recompute the transformed weight from the immutable original weight. Channel rescaling is not repeated during an output refit with fixed codes.

INT4 balancing and centering. For INT4, we compute activation-channel RMS values after per-token absmax normalization, and weight-row RMS values after per-output-channel absmax normalization. These normalizations are used only to estimate balancing statistics. If the resulting magnitudes are a_k and w_k, the input rescaling is proportional to $\sqrt{a_k/w_k}$, with centered log-scales. Output-block RMS equalization and numerical-range safeguards follow.

We subtract the fitting-set channel mean $\boldsymbol{\mu}$ of the transformed activations before INT4 quantization. Its output correction is $\boldsymbol{\mu}\widehat{\boldsymbol{W}}\boldsymbol{T}_{\mathrm{out}}$, computed from the quantized weight and transformation. During refitting, $\boldsymbol{\mu}\widehat{\boldsymbol{W}}$ is included in the fixed features before multiplying by the candidate $\boldsymbol{T}_{\mathrm{out}}$. After fitting, the resulting correction is precomputed for inference. The original module bias remains fixed; centering introduces no independently fitted bias parameters.

NVFP4 rescaling. NVFP4 uses the same RMS equalization construction as MXFP4, with 16-channel equalization groups to match its scale granularity. The 32-channel transformation blocks and input/output grouping remain as specified in Table 4.

B.3 Evaluation protocols

Let p_0 and p_q be the BF16 and quantized next-token distributions. For sequence i and token position t, both models receive the same prefix $c_{i,t}$. Let $\mathcal{I}$ contain the scored sequence/position pairs and let

v range over vocabulary tokens. We report

$$D(p_0 \| p_q) = \frac{1}{|\mathcal{I}|} \sum_{(i,t)\in\mathcal{I}} \sum_{v} p_0(v \mid c_{i,t}) \log \frac{p_0(v \mid c_{i,t})}{p_q(v \mid c_{i,t})}. \tag{36}$$

KL measures agreement between the full models; the fitting procedure uses the local reconstruction objectives in Section 3. Natural logarithms give nats per token, and BF16 self-KL is zero. Perplexity (PPL) is the exponential of token-weighted mean negative log-likelihood on the scored tokens. Reference-answer PPL uses the original assistant answers, whereas WikiText-2 PPL uses observed raw text. Relative KL reduction is $100(1 - D_{\text{candidate}}/D_{\text{reference}})$.

Only assistant continuation tokens are scored, excluding prompts and padding. For a prompt length P and continuation length G, the scored logits are $P - 1, \ldots, P + G - 2$. Reference-answer PPL instead scores the dataset's assistant-answer tokens. The reported paired-bootstrap intervals use 10,000 draws, seed 20260915. Each draw pools KL sums and scored-token counts over the resampled units before computing relative reduction. Tülu resampling keeps all assistant turns of a conversation together; task-conditioned resampling pairs documents across candidates. Intervals are pointwise, not simultaneous across tasks, and describe these fixed checkpoints; they do not include variation from new calibration draws.

Task-conditioned fidelity and accuracy. Each model uses its own tokenizer, native chat template, and saved BF16 continuations, so KL comparisons are within a model. Qwen3 thinking is disabled. Table 2 reports full-task point estimates: MATH `math_verify` ($N = 5{,}000$) (Hendrycks et al., 2021b), MMLU micro accuracy ($N = 14{,}042$) (Hendrycks et al., 2021a), IFEval prompt strict ($N = 541$) (Zhou et al., 2023), HumanEval pass@1 ($N = 164$) (Chen et al., 2021), and HellaSwag `acc_norm` ($N = 10{,}042$) (Zellers et al., 2019). HumanEval uses sanitized generations (Liu et al., 2023) and the original tests. The task-conditioned KL columns use separate 64-prompt samples with shared BF16 continuations, capped at 512 generated tokens. MMLU contributes only 128 scored continuation tokens in the Ministral and Llama panels, so its KL point estimate is particularly sensitive to sampling. Each Llama arm scores exactly the same histories: 21,540 MATH tokens, 128 MMLU tokens, 21,622 IFEval tokens, 12,752 HumanEval tokens, and 32,768 HellaSwag tokens. HellaSwag uses raw continuation prompts; the other tasks use native chat. MMLU trajectory KL measures the answer-letter continuation, distinct from its accuracy likelihood metric.

The four Qwen3 quantized arms in the INT4/MXFP4 diagnostics share 320 saved benchmark continuations and the same 66 Tülu assistant histories. They use the native chat template with thinking disabled and original end-of-turn tokens. For accuracy evaluation, MATH and HumanEval generation are capped at 1,024 tokens and IFEval at 1,280; MMLU uses answer likelihood and HellaSwag raw continuation likelihood.

Independent real-text perplexity. We use `Salesforce/wikitext`, `wikitext-2-raw-v1`, test revision `b08601e04326c79dfdd32d625aee71d232d685c3` (Merity et al., 2017). We join rows with two newlines, retaining empty rows, and use the same tokenizer without a chat template. We split the text into nonoverlapping 2,048-token blocks, resetting context and dropping the incomplete tail. The Ministral shared cache has 147 full blocks, 300,909 scored tokens, and a discarded 561-token tail.

The four Qwen3 INT4/MXFP4 diagnostic arms score the same 298,862 WikiText-2 real-text tokens. WikiText-2 is used only for PPL.

B.4 TÜLU CHAT FIDELITY

The INT4 and MXFP4 diagnostics below compare BF16, WUSH, and TReQ using saved native-chat histories. KL scores BF16-generated assistant tokens; PPL scores dataset reference-answer tokens. Fitting and held-out conversations are distinct; the 32 internal-validation conversations used for selection are excluded. The MXFP4 held-out reference-answer PPL difference (TReQ minus WUSH) is $+0.027$; its paired 95% interval is $[-0.0002, 0.0710]$.

Llama. On the separate 32 fitting and 32 held-out Tülu conversations, Llama INT4 KL is .14282 and .10632, respectively, corresponding to reductions of 57.31% [53.51, 61.12] and 59.11% [55.21, 63.06]. MXFP4 reductions are -5.83% [-11.05, -0.75] and -2.49% [-6.68, 1.87].

Table 6: Ministral Tülu fidelity. Brackets give paired 95% conversation-bootstrap intervals for KL reduction. BF16 reference-answer PPL is 2.274 on fitting and 3.050 on held-out conversations.

Format	Split	WUSH KL	TReQ KL	KL reduction (%)	WUSH / TReQ PPL
INT4	Fit (32)	.4560	.3239	28.97 [24.61, 32.87]	3.026 / 2.639
	Held-out (32)	.4137	.3001	27.47 [22.94, 31.87]	4.248 / 3.673
MXFP4	Fit (32)	.070588	.066891	5.24 [1.95, 8.55]	2.308 / 2.327
	Held-out (32)	.063132	.059969	5.01 [1.56, 8.23]	3.113 / 3.139

Qwen3. On the 32 fitting and 32 held-out Tülu conversations (66 assistant turns total), Qwen3 INT4 KL falls from .26013/.20716 to .17539/.13149, giving reductions of 32.58% [27.24, 37.50] and 36.53% [32.14, 40.10]. MXFP4 KL instead rises from .03469/.02527 to .03885/.02855, giving reductions of -11.99% [-18.49, -6.07] and -12.97% [-21.17, -3.38]. Reference-answer PPL is 2.448/3.365 for BF16, 2.828/4.122 for INT4 WUSH, 2.635/3.702 for INT4 TReQ, 2.476/3.433 for MXFP4 WUSH, and 2.466/3.398 for MXFP4 TReQ (fit/held-out).

B.5 Task-conditioned fidelity intervals

Tables 7 and 8 give paired intervals under the protocol in Appendix B.3. The MXFP4 intervals

Table 7: Relative task-KL reduction (%) from WUSH to the evaluated INT4 TReQ checkpoints ($G_{\text{in}} = 2$, $G_{\text{out}} = 1$), with paired 95% document-bootstrap intervals.

Dataset	Ministral	Llama	Qwen3
MATH	10.28 [6.87, 13.41]	57.65 [56.08, 59.26]	30.61 [25.53, 35.82]
MMLU	26.24 [-10.48, 51.92]	41.91 [19.52, 60.07]	49.03 [25.69, 66.70]
IFEval	31.42 [29.45, 33.27]	55.53 [53.28, 57.71]	33.50 [31.33, 35.62]
HumanEval	48.76 [41.40, 56.13]	68.69 [66.21, 71.17]	50.89 [44.10, 57.28]
HellaSwag	28.81 [25.46, 31.89]	52.72 [50.40, 54.65]	36.50 [31.57, 40.21]

Table 8: Relative task-KL reduction (%) from WUSH to TReQ for the available MXFP4 diagnostics, with paired 95% document-bootstrap intervals.

Dataset	Llama	Qwen3
MATH	-2.19 [-9.80, 5.34]	-8.22 [-15.60, -1.14]
MMLU	-6.13 [-90.74, 41.22]	-56.86 [-339.81, 45.16]
IFEval	-2.62 [-5.58, 0.23]	-12.35 [-16.84, -8.00]
HumanEval	-19.14 [-33.85, -6.31]	1.15 [-29.59, 24.12]
HellaSwag	-11.28 [-15.91, -7.14]	3.24 [-5.34, 12.04]

support KL regressions for Llama on HumanEval and HellaSwag, and for Qwen3 on MATH and IFEval. The remaining intervals cross zero.

B.6 Fixed-checkpoint output-refitting ablations

This INT4 OFF/ON comparison uses a fixed Tülu-calibrated Ministral checkpoint.

For INT4 WUSH, we add identity output transformations to one Tülu-calibrated checkpoint while preserving its decoded weights and activation quantization. OFF evaluates that augmented checkpoint without refitting; ON starts from the same checkpoint and refits only output transformations. Both arms store the same number and shapes of transformations. Native WUSH itself has no added output transformations.

In the INT4 pair, codes, scales, input transformations $\boldsymbol{T}_{\text{in}}$, quantizer settings, grouping, and all noneligible tensors remain frozen, verified across all 34 blocks. Only Q/K/V/O/down output transformations change; gate/up remain fixed. The refits use 96 fitting conversations and 32 internal-validation conversations, with projected Q/K warm start disabled and coefficients 0.3 for direct Q/K, 0.4 for V,

and 0.2 for O/down. Evaluation uses identical saved BF16 histories for 32 fitting and 32 held-out conversations, excluding internal validation, with 10,000 paired conversation-bootstrap draws for the intervals.

Both OFF and ON execute the same augmented output-transformation path. Native WUSH is evaluated separately because adding an identity output transformation introduces intermediate TF32 rounding. Its fitting/held-out KL is 0.45660/0.41490, compared with 0.45600/0.41369 for OFF and 0.40652/0.37798 for ON. The OFF-versus-native relative KL reductions are 0.13% (95% CI: $[-1.20, 1.52]$) and 0.29% ($[-1.09, 1.77]$). These intervals include zero, whereas both ON-versus-OFF reduction intervals are positive. Reference-answer PPL decreases from 3.026 to 2.859 on fitting conversations and from 4.248 to 4.093 on held-out conversations; the paired PPL differences have 95% intervals $[-0.271, -0.099]$ and $[-0.291, -0.077]$, respectively. These are Tülu reference-answer measurements, separate from WikiText perplexity.

Table 9: Ministral Tülu full-model fixed-checkpoint INT4 control. Fitting-set and held-out KL reductions (%) are relative to OFF, with paired 95% intervals. OFF and ON share codes, scales, and input transformations.

Starting checkpoint	Codec	Fitting reduction / CI	Held-out reduction / CI
WUSH + identity output transformations	INT4	10.85 [9.12, 12.56]	8.63 [7.38, 9.90]

This comparison isolates refitting at a fixed WUSH checkpoint; it does not isolate the construction method. It is not equivalent to removing module refits during full-pipeline construction, where downstream quantized weights can also change.

B.6.1 FP4 OUTPUT REFITTING IN A SINGLE DECODER BLOCK

We separately test output refitting on Ministral decoder block 12 (zero-based) in NVFP4 and MXFP4. All seven Q/K/V/O/gate/up/down projections retain native WUSH's 1×1 input/output grouping. We append identity block-diagonal output transformations and fit only these transformations, keeping native input transformations, FP4 codes, block and global scales, and the activation-quantizer configuration fixed. Native WUSH and its identity-augmented version produce bitwise-identical outputs when replaying the complete decoder block. The full refit includes single-layer output fits for all seven projections, including gate/up, projected and direct Q/K refits, V context refitting, and O/down fits to BF16 outputs, with the coefficients in Table 5.

The 128 complete Tülu conversations comprise 96 fitting conversations (138,209 tokens) and 32 internal-validation conversations (47,419 tokens). Internal validation participates in update selection, so these results are not an independent test-set estimate. We evaluate the same fitted transformations with 4-bit activation quantization (A4) and with BF16 activations and activation quantization disabled (A16); there is no separate A16 fit. Within each format and activation mode, native and refitted WUSH are replayed as a pair on the same cached block inputs. Let $\boldsymbol{y}_t^{\mathrm{BF16}}$ and $\widehat{\boldsymbol{y}}_t$ denote the reference and quantized complete decoder-block outputs for token t, including both residual additions. The reported relative MSE is $\sum_t \|\widehat{\boldsymbol{y}}_t - \boldsymbol{y}_t^{\mathrm{BF16}}\|_2^2 / \sum_t \|\boldsymbol{y}_t^{\mathrm{BF16}}\|_2^2$, summed over all internal-validation tokens.

Table 10: Internal-validation relative MSE for Ministral block 12. MSE columns are in units of 10^{-3}; lower is better. Reductions are relative to the paired native WUSH baseline. These are point estimates without confidence intervals.

Format	Activations	Native WUSH	WUSH + full output refit	Reduction (%)
NVFP4	A4	2.425431	2.115101	12.79
NVFP4	A16	0.819208	0.758084	7.46
MXFP4	A4	3.299695	3.033627	8.06
MXFP4	A16	1.058172	0.998701	5.62

For this block and these two formats and activation modes, output refitting reduces reconstruction error by 5.62–12.79% without additional input-transform groups or requantization. The reported output-refitting experiments do not isolate full channel mixing from diagonal output scaling.

B.7 STORAGE AND CALIBRATION COST

The Ministral Tülu storage accounting covers 238 projections and 7,415,529,472 original weight entries. The accounting scope is the quantized transformer-block linear projections; embedding, head, normalization, and vision parameters are excluded.

INT4 construction cost. Saved single-GPU worker measurements cover all 34 Ministral, 32 Llama, and 36 Qwen3 blocks on GH200. The summed worker times are 3.09, 4.91, and 4.29 hours, respectively; median block times are 359, 570, and 452 seconds. The timings include construction, module refits, packing, export/reload, checkpoint hashing, and local parity and reconstruction checks. They exclude initial model and cache loading and the preceding moment/input-cache capture. Because blocks run independently, these sums are aggregate worker time rather than end-to-end elapsed time. The maximum process-level allocated GPU memory across workers is 66.04, 68.13, and 61.56 GiB, respectively, including allocations before the timed construction interval.

Storage accounting. For a $d_{\text{in}} \times d_{\text{out}}$ MXFP4 projection, assume two FP4 codes per byte, one E8M0 scale byte per 32 weights, BF16 input transformations, and output transformations counted at 19 bits per TF32 value. With G_{out} output groups and G_{in} input groups, the corresponding layout formula is

$$\text{BPE}_q = 4 + \frac{8}{32} + \frac{512 G_{\text{out}}}{d_{\text{out}}} + \frac{608 G_{\text{in}}}{d_{\text{in}}}. \tag{37}$$

This formula describes the compact MXFP4 payload; expanded fake-quant codes and FP32 scale containers are not counted as deployed FP4/E8M0 storage. The TReQ input and output transformation sizes are read from the current checkpoint, without assuming the same grouping for every projection. If B_q is the summed payload in bytes and P_q the original quantized weight count,

$$\text{BPE} = \frac{8 B_q}{P_q}. \tag{38}$$

Bytes are summed across projections before normalization, counting all stored transformations and any duplicated group copies without assuming sharing or extra compression. INT4 counts packed 4-bit codes, FP32 per-output-channel scales for each tile, BF16 input transforms, TF32 output transformations, and FP32 activation means and static bias buffers. Zero mean/bias buffers present in the runtime are included; unused identity output transformations in native WUSH's bypass path are excluded. The MXFP4 WUSH + refit entry is calculated from the identity-augmented layout, with one TF32 32×32 output-transformation block per 32 output channels. Refitting changes values, not payload size; this storage calculation is independent of accuracy or fidelity measurements.

For NVFP4, we count packed 4-bit E2M1 codes, one E4M3 scale byte per 16 weights, BF16 input transforms, TF32 output transformations, and FP32 global scales. Transform shapes and group multiplicities are taken from the evaluated checkpoints. For their stored layouts,

$$\text{BPE}_{\text{WUSH,NV}} = 4 + \frac{8}{16} + \frac{512}{d_{\text{out}}} + \frac{32}{d_{\text{in}} d_{\text{out}}}, \tag{39}$$

$$\text{BPE}_{\text{TReQ,NV}} = 4 + \frac{8}{16} + \frac{512 G_{\text{out}}}{d_{\text{out}}} + \frac{608 G_{\text{in}}}{d_{\text{in}}} + \frac{32 G_{\text{out}}(G_{\text{in}} + 1)}{d_{\text{in}} d_{\text{out}}}. \tag{40}$$

WUSH stores one global weight scale per projection; TReQ stores $G_{\text{out}} G_{\text{in}}$ weight scales and G_{out} activation scales. Table 2 gives the resulting BPE in each datatype header, in WUSH/TReQ order.

Table 11: Quantized-projection weight counts and compact payloads in bytes. Output transformations count 19 bits per TF32 value.

Model	Codec	Weight count	WUSH payload	TReQ payload
Ministral	INT4	7,415,529,472	3,809,427,456	4,037,541,888
Ministral	MXFP4	7,415,529,472	4,024,172,544	4,177,920,000
Ministral	NVFP4	7,415,529,472	4,255,908,792	4,409,658,832
Llama	NVFP4	6,979,321,856	4,005,561,216	4,150,267,136
Qwen3	NVFP4	6,945,767,424	3,991,929,840	4,143,517,344

Adding output transformations to WUSH gives 4.2296 BPE for INT4 and 4.4612 BPE for MXFP4, unchanged between refit OFF and ON. Tensor payload is distinct from fake-quantization checkpoint disk size, allocator alignment, file headers, KV cache, and temporary workspace. The evaluator's decoded weights and intermediate tensors are not intrinsic deployment storage. A fused implementation can consume grouped partials on chip before writing the output; calibration normal systems require separate temporary memory.

B.8 Native kernel performance on B200

Paired GEMM protocol. The measurements use an NVIDIA B200 with CUDA 13.3, PyTorch 2.12.1+cu130, and CUTLASS 4.3.0. Seven variants share the same packed E2M1 code buffers and E8M0 scale buffers: plain and fused (TF32 and FP16) one-SM $128 \times 128 \times 256$ tiles, plain and fused two-SM $256 \times 128 \times 256$ tiles, and PyTorch's `scaled_mm` with a two-SM $256 \times 256 \times 128$ tile. Tile dimensions are $(n, d_{\text{out}}, d_{\text{in}})$. Only the fused variants apply the block-diagonal output transformation. The plain variants produce bit-identical BF16 GEMM outputs; fused variants of the same precision match each other and a separate transform kernel of that precision bitwise in these microbenchmarks.

Each configuration uses at least ten round-robin warm-ups and 30 trials of CUDA Graphs of 20 calls, rotating variant order between trials. Table 12 gives median per-call times (mean of two runs). Offline weight packing and transform preparation are excluded. Inputs are already quantized, so these times also exclude input transformation and activation quantization. The GPU was power-limited at approximately 1,000 W, with reported SM clocks of 1,600–1,965 MHz; absolute times varied by about 10% across runs. Interpret the paired measurements within each row. Replays can retain part of the working set in the 126 MB L2 cache. Derived ratios in this paper are approximate, computed from the unrounded latencies.

Full-model prefill. The native operator runs in Transformers 5.9.0 with a Ministral-3-8B MXFP4/A4 checkpoint containing 34 blocks and 238 projections with trained 32×32 input and output transformations. Of these, 102 Q/K/V projections use $G_{\text{out}} = G_{\text{in}} = 2$: input transformations are indexed by output halves, and output transformations by input (reduction) halves. Their grouped GEMM partials are transformed and summed in FP32 on chip before one BF16 rounding. The other 136 projections use ungrouped operators. Embeddings, norms, and the head are BF16. The native and BF16 runs use the same GPU, inputs, and cuDNN SDPA attention backend; BF16 projections use cuBLAS.

We time GPU-resident token IDs through last-token logits, with a fresh KV cache. Each entry in Table 13 is the median of 20 forwards after five warm-ups, without CUDA Graphs or compilation. These timings include input transformations, dynamic activation quantization, attention, and the remaining model operations. The no-$\boldsymbol{T}_{\text{out}}$ control retains the same checkpoint and input-side computation but drops the output transformations, using either the paired plain GEMM or PyTorch's GEMM. It computes a different function and is not a separately constructed WUSH model. Paired controls and fused runs were executed back to back; repeated fused timings agreed to 2.2%. The operator dispatches to the two-SM tile from $n = 256$.

At batch 1 and length 8,192, profiling attributes 32.0 ms to the native linear operators, including input transformations and quantization, versus 71.4 ms for BF16 GEMMs. Approximately 76 ms of attention, normalization, rotary embedding, residual, and other elementwise work is shared. These separately profiled components need not sum to the unprofiled median latency. At batch 1 and length 128, the native model issues 680 launches per forward versus 240 cuBLAS calls; host-launch cost limits the short-sequence result. Table 14 retains the projection-level operator measurements, including grouped Q/K, which are distinct from the prequantized GEMM microbenchmark.

FP16 output transformation. The FP16 variant rounds the FP32 product and $\boldsymbol{T}_{\text{out}}$ to FP16 and applies the transformation with FP16 tensor-core MMAs accumulating in FP32. FP16 has the same 10-bit significand as TF32, so rounded values agree within the FP16 normal range $[2^{-14}, 65504]$; larger magnitudes overflow. The narrower operands halve the transformation MMAs per tile. On 4,096 tokens of English text, the largest product magnitude over the 136 ungrouped Ministral projections is 763.5, with no overflow; grouped Q/K/V projections keep TF32 in the FP16 model. On

Table 12: Paired MXFP4 GEMM latency on B200 in μs. Columns marked $+\boldsymbol{T}_{\text{out}}$ add the TF32 or FP16 output transformation to the same tile. PyTorch applies no output transformation. Every supplied microbenchmark shape is shown.

			1-SM				2-SM		
d_{out}	d_{in}	n	plain	$+\boldsymbol{T}_{\text{out}}$ TF32	$+\boldsymbol{T}_{\text{out}}$ FP16	PyTorch	plain	$+\boldsymbol{T}_{\text{out}}$ TF32	$+\boldsymbol{T}_{\text{out}}$ FP16
4096	4096	1	7.1	7.4	7.1	7.1	9.5	9.8	9.6
4096	4096	16	6.7	6.9	6.8	7.1	7.4	7.6	7.5
4096	4096	64	6.6	6.9	6.8	7.0	7.4	7.7	7.6
4096	4096	256	6.9	7.2	6.9	10.1	6.6	6.9	6.8
4096	4096	1024	10.8	11.2	11.1	11.1	10.1	10.4	10.2
4096	4096	2048	18.2	18.6	18.3	17.0	16.3	16.9	16.7
4096	4096	4096	33.5	34.0	33.1	31.7	28.7	30.3	30.1
4096	4096	8192	70.3	70.7	70.5	66.7	57.9	59.8	59.0
4096	4096	16384	139.8	141.4	141.2	134.4	113.7	116.2	114.7
8192	8192	256	10.1	10.3	10.2	11.2	9.4	9.7	9.7
8192	8192	1024	33.2	32.9	32.7	28.4	29.0	29.2	29.4
8192	8192	4096	132.0	132.5	132.2	109.2	107.3	109.7	108.5
8192	8192	8192	253.5	255.5	250.2	260.7	205.9	213.8	207.1
14336	4096	256	10.8	11.2	11.0	10.9	9.7	10.0	9.8
14336	4096	1024	30.7	32.1	31.4	31.2	26.8	28.3	27.3
14336	4096	4096	122.5	123.0	126.1	147.4	101.9	102.9	102.4
4096	14336	256	13.7	14.0	13.8	22.0	12.5	12.8	12.7
4096	14336	1024	27.6	27.9	27.6	25.1	24.0	24.4	24.2
4096	14336	4096	108.6	109.1	108.8	101.0	89.7	91.3	90.5
12288	4096	256	10.6	10.9	10.7	10.6	9.4	9.9	9.6
12288	4096	1024	26.4	26.8	26.5	24.1	22.7	23.7	23.3
12288	4096	4096	106.1	106.0	107.4	126.1	88.6	89.1	88.7

Table 13: Ministral-3-8B full-model prefill on B200. Latencies are in ms; memory is peak GiB (BF16/native). The two no-$\boldsymbol{T}_{\text{out}}$ columns are timing-only controls. The added cost is $100(t_{\text{fused}}/t_{\text{plain}} - 1)$, using unrounded latencies. In the FP16 column, grouped Q/K/V projections retain TF32 output transformations.

				Fused			Cost (%)		
Batch	Length	BF16	Plain, no $\boldsymbol{T}_{\text{out}}$	TF32	FP16	PyTorch, no $\boldsymbol{T}_{\text{out}}$	TF32	FP16	Memory
1	128	10.9	12.9	13.5	13.3	19.2	4.8	3.3	15.9 / 6.8
1	512	14.1	13.3	13.6	13.5	19.2	2.2	1.7	16.0 / 6.9
1	2048	36.7	29.2	29.6	29.6	32.9	1.3	1.1	16.3 / 7.2
1	4096	74.0	53.1	53.4	53.3	61.5	0.4	0.3	16.8 / 7.7
1	8192	151.2	108.1	108.5	108.3	121.5	0.3	0.1	17.8 / 8.7
4	128	14.0	13.1	13.5	13.4	19.0	2.7	2.5	16.0 / 6.9
4	512	36.6	29.1	29.5	29.5	32.8	1.5	1.3	16.3 / 7.2
4	2048	143.4	100.2	101.0	100.8	113.5	0.8	0.6	17.8 / 8.7
4	4096	288.5	199.3	200.6	200.1	226.7	0.7	0.4	19.8 / 10.7
4	8192	600.7	414.6	415.6	415.1	463.9	0.2	0.1	23.7 / 14.6

Table 14: Projection operator timings at $n = 8192$ in μs. Grouped Q/K use 2×2 input/output groups and keep TF32 in the FP16 column. All entries use the same projection-specific input-side computation for their paired controls.

Projection	d_{in}	d_{out}	Plain, no $\boldsymbol{T}_{\text{out}}$	Fused TF32	Fused FP16	PyTorch, no $\boldsymbol{T}_{\text{out}}$
q_proj	4096	4096	103.4	111.2	111.8	165.6
k_proj	4096	1024	70.3	73.5	73.6	90.1
o_proj	4096	4096	74.9	76.8	76.3	103.6
gate_proj	4096	14336	215.3	219.2	218.8	329.1
down_proj	14336	4096	249.6	253.6	251.2	307.4

the two-SM tile, FP16 reduces the added time at most shapes with $n \geq 1024$, often by about half (Table 12); in prefill, FP16 is faster by at most 0.5 ms per forward (Table 13).

Decode operator. For decoding with $n \leq 16$ rows, a separate W4A8 operator shares the packed MXFP4 weight and scale buffers of the prefill operator. One kernel applies the input transformation in BF16 with FP32 accumulation and quantizes the result to MXFP8: E4M3 elements with an E8M0 scale $2^{\lceil \log_2(a/448) \rceil}$ per 32 entries along d_{in}, where a is the block maximum magnitude. A second kernel computes the block-scaled product with `tcgen05` MMAs in which the weight is the 128-row operand and the rows are padded to 16. d_{in} is split across the CTAs of a thread-block cluster. The owner of each 32-column output block sums the FP32 partial products, rounds them to TF32 or FP16, applies $\boldsymbol{T}_{\text{out}}$ and the bias, and rounds once to BF16. The paired control is the same kernel with the same split configuration but without the rounding and $\boldsymbol{T}_{\text{out}}$; it computes a different function. Each entry of Table 15 is the median of 20 trials of a CUDA Graph over 24 weight copies, so that every call reads its weights from HBM; variant order rotates between trials. Over $n \in \{1, 4, 8, 16\}$ and the five Ministral projection shapes, output transformations add 0.23–0.42 μs to the GEMM kernel and 2.8–8.6% to the operator; grouped Q/K/V projections apply separate output transformations to the two input-group partial products. With output transformations, the operator is 1.02–1.78$\times$ faster than the BF16 matrix multiplication. Its outputs match an FP64 reference computed from the operator's own MXFP8 codes to within one BF16 ulp plus one TF32 or FP16 ulp of the partial products. The evaluated models quantize activations to four bits; model quality with MXFP8 activations is not measured.

Table 15: Paired W4A8 decode operator latency on B200 in μs. GEMM columns time the decode GEMM kernel alone; operator columns add input transformation and MXFP8 quantization. No-$\boldsymbol{T}_{\text{out}}$ columns are timing-only controls. BF16 is a cuBLAS matrix multiplication of the same shape. †Two input groups and two output groups.

				GEMM			Operator			
Projection	d_{in}	d_{out}	n	no $\boldsymbol{T}_{\text{out}}$	TF32	FP16	no $\boldsymbol{T}_{\text{out}}$	TF32	FP16	BF16
q_proj†	4096	4096	1	5.24	5.66	5.58	7.18	7.55	7.53	8.83
q_proj†	4096	4096	16	5.33	5.68	5.67	7.20	7.63	7.56	11.14
k/v_proj†	4096	1024	1	3.64	4.05	3.89	5.93	6.44	6.28	6.56
k/v_proj†	4096	1024	16	3.72	4.06	3.98	6.11	6.49	6.44	6.63
o_proj	4096	4096	1	5.33	5.58	5.58	7.02	7.22	7.26	8.99
o_proj	4096	4096	16	5.33	5.59	5.59	7.12	7.33	7.38	11.37
gate/up_proj	4096	14336	1	9.17	9.54	9.58	11.32	11.82	11.73	20.22
gate/up_proj	4096	14336	16	9.35	9.73	9.77	11.56	11.99	11.99	20.22
down_proj	14336	4096	1	9.01	9.25	9.26	11.89	12.33	12.41	21.92
down_proj	14336	4096	16	9.36	9.61	9.59	11.82	12.31	12.32	21.60

Numerical scope. Seven real-weight projection fixtures have relative MSE at most 1.1×10^{-9} against the reference. With all 238 projections evaluated on the reference execution's own inputs, median relative MSE is 5×10^{-10}. These local checks do not imply full-model logit equivalence. On a 64-random-token diagnostic, the GPU reference and native model have 62.5% top-1 agreement and reported mean KL 0.073 nats (maximum 0.32). CPU and GPU reference executions themselves have 62.5% agreement and mean KL 0.090 (maximum 0.43); the CPU reference versus native model gives 56.2% and 0.096 (maximum 0.53). With FP16 output transformations, the GPU reference and native model have 59.4% top-1 agreement and mean KL 0.084 (maximum 0.42). These diagnostics indicate sensitivity to numerical execution order, but do not establish downstream accuracy parity or identify the cause of every discrepancy. The task-quality results in Table 2 use the separate quantize/dequantize backend; they were not remeasured with the native kernel. No INT4 throughput or end-to-end autoregressive decoding speed is measured here.

B.9 COMPLETE HELD-OUT BLOCK RECONSTRUCTION RESULTS

Table 16 contains all 306 model/block/format comparisons under the protocol of Section 4.2. The all-block averages include every layer. MSE increases at Llama blocks 8 and 14 in MXFP4, Qwen3 blocks 0, 4, and 5 in MXFP4, and Qwen3 blocks 0 and 4 in NVFP4. A few large relative improvements affect the means: for example, median MXFP4 changes are -3.48% for Llama and -2.05% for Qwen3, compared with means of -7.06% and -6.61%.

Table 16: Complete held-out block MSE changes (%) from WUSH to TReQ. Negative values indicate reductions; positive values indicate increases. All formats use A4W4. Dashes denote blocks absent from the model. Avg. is the equally weighted mean across all blocks, computed before rounding.

Block	Ministral-3-8B			Llama-3.1-8B			Qwen3-8B		
	INT4	MXFP4	NVFP4	INT4	MXFP4	NVFP4	INT4	MXFP4	NVFP4
0	−72.16%	−26.46%	−16.75%	−40.16%	−4.01%	−2.64%	−45.92%	+0.90%	+0.84%
1	−37.30%	−20.58%	−23.35%	−94.76%	−99.58%	−55.36%	−40.66%	−3.40%	−25.86%
2	−96.76%	−27.10%	−24.90%	−46.31%	−16.50%	−14.41%	−39.09%	−22.10%	−43.20%
3	−31.82%	−44.82%	−52.22%	−59.26%	−19.37%	−18.50%	−33.45%	−1.38%	−11.84%
4	−39.70%	−53.11%	−33.94%	−51.25%	−6.88%	−8.16%	−35.72%	+15.81%	+10.73%
5	−25.64%	−27.26%	−28.54%	−53.77%	−3.91%	−2.65%	−29.49%	+3.07%	−0.48%
6	−36.77%	−36.86%	−52.93%	−42.02%	−3.97%	−2.35%	−96.34%	−99.91%	−99.90%
7	−34.65%	−28.30%	−36.69%	−56.12%	−0.66%	−0.76%	−20.80%	−0.40%	−6.12%
8	−26.37%	−18.74%	−30.95%	−38.35%	+0.18%	−0.35%	−21.95%	−2.07%	−5.23%
9	−37.90%	−25.80%	−18.29%	−44.20%	−6.32%	−3.49%	−22.04%	−2.11%	−3.05%
10	−26.48%	−45.14%	−33.92%	−57.97%	−3.85%	−2.64%	−26.31%	−2.55%	−4.24%
11	−28.07%	−21.19%	−36.82%	−48.39%	−1.60%	−1.51%	−25.88%	−0.78%	−2.46%
12	−27.44%	−13.38%	−16.25%	−37.45%	−1.16%	−0.92%	−26.86%	−1.45%	−3.53%
13	−29.84%	−19.50%	−16.63%	−41.57%	−1.77%	−1.28%	−24.53%	−1.46%	−2.02%
14	−28.44%	−18.75%	−30.60%	−47.25%	+0.29%	−0.95%	−24.81%	−0.70%	−2.23%
15	−27.16%	−12.15%	−9.61%	−48.16%	−1.59%	−1.83%	−22.57%	−0.32%	−2.77%
16	−31.50%	−12.25%	−13.38%	−52.55%	−0.87%	−1.63%	−49.49%	−5.43%	−32.61%
17	−30.14%	−13.69%	−18.10%	−50.78%	−1.95%	−2.08%	−23.63%	−2.69%	−3.03%
18	−49.40%	−47.38%	−40.75%	−66.96%	−2.75%	−2.83%	−25.53%	−0.63%	−1.69%
19	−31.99%	−12.20%	−13.68%	−68.18%	−2.92%	−5.73%	−27.37%	−1.75%	−5.92%
20	−32.61%	−10.33%	−10.31%	−73.57%	−4.55%	−5.62%	−23.24%	−1.31%	−3.43%
21	−37.12%	−11.80%	−12.32%	−61.15%	−3.07%	−5.03%	−26.33%	−1.26%	−4.29%
22	−36.04%	−10.80%	−14.15%	−69.45%	−2.89%	−3.25%	−28.17%	−2.04%	−5.05%
23	−31.54%	−11.14%	−14.41%	−71.87%	−4.69%	−6.16%	−27.95%	−0.14%	−3.35%
24	−29.59%	−12.82%	−15.78%	−57.55%	−5.97%	−7.53%	−28.57%	−1.61%	−5.99%
25	−30.66%	−10.08%	−11.49%	−67.13%	−6.69%	−6.89%	−25.80%	−2.66%	−6.45%
26	−36.02%	−12.27%	−14.84%	−59.81%	−3.74%	−5.46%	−27.19%	−3.77%	−5.85%
27	−43.72%	−26.60%	−17.73%	−71.18%	−5.33%	−5.27%	−27.11%	−3.20%	−6.49%
28	−31.54%	−14.88%	−14.97%	−60.62%	−4.41%	−4.60%	−25.46%	−1.43%	−3.75%
29	−33.84%	−18.33%	−20.97%	−53.66%	−3.21%	−6.28%	−30.57%	−5.09%	−7.38%
30	−30.69%	−15.55%	−18.15%	−39.74%	−0.93%	−3.26%	−30.73%	−6.88%	−8.12%
31	−49.44%	−12.75%	−19.97%	−42.56%	−1.14%	−4.53%	−29.14%	−4.66%	−5.46%
32	−35.03%	−18.60%	−16.45%	—	—	—	−22.24%	−4.65%	−5.84%
33	−56.33%	−23.62%	−38.50%	—	—	—	−33.62%	−5.23%	−7.84%
34	—	—	—	—	—	—	−34.12%	−15.71%	−11.11%
35	—	—	—	—	—	—	−77.20%	−49.06%	−4.44%
Avg.	−37.17%	−21.59%	−23.19%	−55.43%	−7.06%	−6.06%	−32.22%	−6.61%	−9.43%

C TWO-SIDED WUSH INITIALIZATION AND DERIVATION

The input-side construction follows WUSH (Chen et al., 2026); applying its inverse-transpose pair to an activation-weighted output moment gives our two-sided initialization. We use $\boldsymbol{W}_0 \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ and block size $b = 32$. The derivation assumes exact arithmetic and positive-definite moments; the final subsection specifies the numerical safeguards and recomputation for stored factors.

C.1 AN INVERSE-TRANSPOSE PAIR FROM TWO MOMENTS

Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^b$ be column vectors with second moments $\boldsymbol{M}_A$ and $\boldsymbol{M}_B$. These are uncentered moments; no zero-mean assumption is needed. We seek transforms that redistribute their energies while preserving their inner product:

$$(\boldsymbol{T}_A \boldsymbol{a})^\top (\boldsymbol{T}_B \boldsymbol{b}) = \boldsymbol{a}^\top \boldsymbol{b}, \qquad \boldsymbol{T}_A^\top \boldsymbol{T}_B = \boldsymbol{I}. \tag{41}$$

Write Cholesky factorizations $\boldsymbol{M}_A = \boldsymbol{C}_A \boldsymbol{C}_A^\top$ and $\boldsymbol{M}_B = \boldsymbol{C}_B \boldsymbol{C}_B^\top$, and take the SVD $\boldsymbol{C}_A^\top \boldsymbol{C}_B = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^\top$. For a normalized Hadamard matrix $\boldsymbol{R}$ and $0 \le \beta \le 1/2$, define

$$\boldsymbol{T}_A = \boldsymbol{R} \boldsymbol{\Sigma}^{1-2\beta} \boldsymbol{U}^\top \boldsymbol{C}_A^{-1} = \boldsymbol{R} \boldsymbol{\Sigma}^{-2\beta} \boldsymbol{V}^\top \boldsymbol{C}_B^\top, \tag{42}$$

$$\boldsymbol{T}_B = \boldsymbol{R} \boldsymbol{\Sigma}^{2\beta} \boldsymbol{V}^\top \boldsymbol{C}_B^{-1} = \boldsymbol{R} \boldsymbol{\Sigma}^{2\beta-1} \boldsymbol{U}^\top \boldsymbol{C}_A^\top. \tag{43}$$

The equal forms follow from the SVD. In particular,

$$\boldsymbol{T}_A^\top \boldsymbol{T}_B = \boldsymbol{C}_A^{-\top} \boldsymbol{U} \boldsymbol{\Sigma}^{1-2\beta} \boldsymbol{\Sigma}^{2\beta-1} \boldsymbol{U}^\top \boldsymbol{C}_A^\top = \boldsymbol{I}. \tag{44}$$

The transformed moments are

$$\boldsymbol{T}_A \boldsymbol{M}_A \boldsymbol{T}_A^\top = \boldsymbol{R} \boldsymbol{\Sigma}^{2-4\beta} \boldsymbol{R}^\top, \qquad \boldsymbol{T}_B \boldsymbol{M}_B \boldsymbol{T}_B^\top = \boldsymbol{R} \boldsymbol{\Sigma}^{4\beta} \boldsymbol{R}^\top. \tag{45}$$

Thus β allocates the singular-value scaling between the two operands. At $\beta = 1/4$, both moments equal $\boldsymbol{R} \boldsymbol{\Sigma} \boldsymbol{R}^\top$. Since every squared Hadamard entry is $1/b$, every diagonal entry of this common moment equals $\operatorname{tr}(\boldsymbol{\Sigma})/b$. This equalizes marginal second moments, without requiring off-diagonal correlations to vanish or bounding individual activation outliers.

A balancing interpretation. The symmetric split minimizes a sum-of-energies surrogate over all invertible inverse-transpose pairs:

$$J(\boldsymbol{T}) = \operatorname{tr}(\boldsymbol{T} \boldsymbol{M}_A \boldsymbol{T}^\top) + \operatorname{tr}(\boldsymbol{T}^{-\top} \boldsymbol{M}_B \boldsymbol{T}^{-1}) \;\ge\; 2 \operatorname{tr}(\boldsymbol{\Sigma}). \tag{46}$$

To see this, set $\boldsymbol{F} = \boldsymbol{T} \boldsymbol{C}_A \boldsymbol{U}$ and $\boldsymbol{G} = \boldsymbol{T}^{-\top} \boldsymbol{C}_B \boldsymbol{V}$. Then $\boldsymbol{F}^\top \boldsymbol{G} = \boldsymbol{\Sigma}$, so $J(\boldsymbol{T}) - 2 \operatorname{tr}(\boldsymbol{\Sigma}) = \|\boldsymbol{F} - \boldsymbol{G}\|_F^2 \ge 0$. For the pair above with $\beta = 1/4$, $\boldsymbol{F} = \boldsymbol{G} = \boldsymbol{R} \boldsymbol{\Sigma}^{1/2}$, attaining the bound. This explains moment balancing; J is not the deployed activation-and-weight quantization loss.

C.2 APPLYING THE PAIR ON THE INPUT SIDE

Let $\boldsymbol{A}_c$ contain n_c activation vectors from this projection's input in the original BF16 model. Define

$$\boldsymbol{H}_{\text{raw}} = \frac{1}{n_c} \boldsymbol{A}_c^\top \boldsymbol{A}_c, \qquad \boldsymbol{H}_{\text{in}} = \frac{\boldsymbol{H}_{\text{raw}}}{\operatorname{tr}(\boldsymbol{H}_{\text{raw}})/d_{\text{in}}}. \tag{47}$$

This input second moment has mean diagonal one. For squared linear reconstruction error, the unnormalized second moment is proportional to the weight Hessian; it is computed from forward activations. For input block i, let $\boldsymbol{W}_i$ contain the corresponding 32 rows of $\boldsymbol{W}_0$, and define

$$\boldsymbol{M}_{X,i} = \boldsymbol{H}_{\text{in},ii}, \qquad \boldsymbol{M}_{W,i} = \frac{1}{d_{\text{out}}} \boldsymbol{W}_i \boldsymbol{W}_i^\top. \tag{48}$$

Applying Eqs. (42)–(43) gives an auxiliary matrix $\boldsymbol{L}_i = \boldsymbol{T}_A$ (with $\boldsymbol{M}_A = \boldsymbol{M}_{W,i}$ and $\boldsymbol{M}_B = \boldsymbol{M}_{X,i}$). Thus $\boldsymbol{L}_i^{-1} = \boldsymbol{T}_B^\top$ before numerical safeguards. We set

$$\boldsymbol{T}_{\text{in},i} = \boldsymbol{L}_i^{-1}, \qquad \widetilde{\boldsymbol{W}}_i = \boldsymbol{L}_i \boldsymbol{W}_i. \tag{49}$$

For a row activation $\boldsymbol{a}_i^\top$ and a weight column $\boldsymbol{w}_i$, the transformed product is

$$(\boldsymbol{a}_i^\top \boldsymbol{T}_{\text{in},i})(\boldsymbol{L}_i \boldsymbol{w}_i) = \boldsymbol{a}_i^\top \boldsymbol{w}_i. \tag{50}$$

Assembling $\boldsymbol{L} = \operatorname{bdiag}(\boldsymbol{L}_i)$ gives $\boldsymbol{T}_{\text{in}} = \boldsymbol{L}^{-1}$ and $\widetilde{\boldsymbol{W}} = \boldsymbol{L} \boldsymbol{W}_0$, hence $\boldsymbol{T}_{\text{in}} \widetilde{\boldsymbol{W}} = \boldsymbol{W}_0$. At the symmetric split, Eq. (45) balances the weight-row moment and the transformed activation moment within each input block.

C.3 Extending the construction to the output side

The second stage takes account of the energies of the transformed input channels. Define

$$\boldsymbol{H}' = \boldsymbol{T}_{\text{in}}^{\top} \boldsymbol{H}_{\text{in}} \boldsymbol{T}_{\text{in}}, \qquad \boldsymbol{e} = \operatorname{diag}(\boldsymbol{H}'), \qquad \boldsymbol{D}_e = \operatorname{diag}(\boldsymbol{e}). \tag{51}$$

The transformed activation energy needed by the output-side construction is the concatenation of

$$\boldsymbol{e}_i = \operatorname{diag}\left(\boldsymbol{T}_{\text{in},i}^{\top} \boldsymbol{H}_{\text{in},ii} \boldsymbol{T}_{\text{in},i}\right). \tag{52}$$

For output block j, let $\boldsymbol{B}_j = \widetilde{\boldsymbol{W}}[:, bj : b(j+1)]$ and use

$$\boldsymbol{M}_{R,j} = \frac{1}{d_{\text{out}}} \boldsymbol{B}_j^{\top} \boldsymbol{D}_e \boldsymbol{B}_j. \tag{53}$$

This is the column Gram matrix of $\boldsymbol{D}_e^{1/2} \boldsymbol{B}_j / \sqrt{d_{\text{out}}}$. It weights each transformed input channel by its second moment. Replacing $\boldsymbol{H}'$ by $\boldsymbol{D}_e$ drops cross-channel correlations; consequently $\boldsymbol{M}_{R,j}$ is a diagonal-moment surrogate, not the full output second moment $\boldsymbol{B}_j^{\top} \boldsymbol{H}' \boldsymbol{B}_j$ (nor a centered covariance).

Apply the generic pair to $(\boldsymbol{M}_{R,j}, \boldsymbol{I})$, obtaining $(\boldsymbol{R}_j, \boldsymbol{P}_j)$ with $\boldsymbol{R}_j^{\top} \boldsymbol{P}_j = \boldsymbol{I}$. The identity is the reference metric for the initially unchanged output coordinates. For each row $\boldsymbol{b}^{\top}$ of $\boldsymbol{B}_j$ and each coordinate vector $\boldsymbol{u}$, Eq. (41) gives

$$(\boldsymbol{b}^{\top} \boldsymbol{R}_j^{\top})(\boldsymbol{P}_j \boldsymbol{u}) = \boldsymbol{b}^{\top} \boldsymbol{u}. \tag{54}$$

We therefore take

$$\boldsymbol{W}_{\text{mid},j} = \boldsymbol{B}_j \boldsymbol{R}_j^{\top}, \qquad \boldsymbol{T}_{\text{out},j} = \boldsymbol{P}_j = \boldsymbol{R}_j^{-\top}. \tag{55}$$

At $\beta = 1/4$, before numerical safeguards, the pair balances

$$\frac{1}{d_{\text{out}}} \boldsymbol{W}_{\text{mid},j}^{\top} \boldsymbol{D}_e \boldsymbol{W}_{\text{mid},j} = \boldsymbol{R}_j \boldsymbol{M}_{R,j} \boldsymbol{R}_j^{\top} = \boldsymbol{P}_j \boldsymbol{P}_j^{\top}. \tag{56}$$

The two sides describe the weighted transformed weight columns and the output-transformation rows. The Hadamard factor equalizes their marginal energies. The output transformation need not be orthogonal.

C.4 Exact factorization and numerical scope

Let $\boldsymbol{R}_{\text{out}} = \operatorname{bdiag}(\boldsymbol{R}_j)$ and $\boldsymbol{P}_{\text{out}} = \operatorname{bdiag}(\boldsymbol{P}_j)$. The two stages yield

$$\boldsymbol{T}_{\text{in}} = \boldsymbol{L}^{-1}, \qquad \boldsymbol{W}_{\text{mid}} = \boldsymbol{L} \boldsymbol{W}_0 \boldsymbol{R}_{\text{out}}^{\top}, \qquad \boldsymbol{T}_{\text{out}} = \boldsymbol{P}_{\text{out}}, \tag{57}$$

and therefore

$$\boldsymbol{T}_{\text{in}} \boldsymbol{W}_{\text{mid}} \boldsymbol{T}_{\text{out}} = \boldsymbol{L}^{-1} \boldsymbol{L} \boldsymbol{W}_0 \boldsymbol{R}_{\text{out}}^{\top} \boldsymbol{P}_{\text{out}} = \boldsymbol{W}_0. \tag{58}$$

For grouped projections, the same argument applies to each weight submatrix using its local dimensions and moments; summing the input-group contributions recovers the original linear map.

For the practical construction, replace each positive-semidefinite moment $\boldsymbol{M}$ by

$$\frac{\boldsymbol{M} + \boldsymbol{M}^{\top}}{2} + 10^{-4} \max\left(\frac{|\operatorname{tr} \boldsymbol{M}|}{b}, \epsilon_{64}\right) \boldsymbol{I}, \tag{59}$$

where ϵ_{64} is FP64 machine epsilon, and floor the singular values at ϵ_{64}. When the epsilon floor is inactive, damping scales proportionally with the moment. Both input and output splits use $\beta = 1/4$ for INT4 and MXFP4 and $\beta = 5/16$ for NVFP4. Damping preserves the algebra for the resulting positive-definite moments, although it changes the moments being balanced; in particular, damping an identity reference introduces an isotropic scale. Singular-value flooring and storage rounding can break the exact inverse-pair identities. For stored invertible transformations, we restore the continuous target by recomputing $\boldsymbol{W}_{\text{mid}} = \boldsymbol{T}_{\text{in}}^{-1} \boldsymbol{W}_0 \boldsymbol{T}_{\text{out}}^{-1}$, as in Eq. (4). This preserves the factorization in exact arithmetic for the stored factors, subject to numerical solve error. Quantizing this transformed weight and rounding the deployed operations then introduce error. Subsequent single-layer and module-level output refits use the reconstruction objectives in Section 3.