SHAMAN-Q: SHAMPOO AUGMENTED NANOQUANT FOR SUB-1-BIT LLM WEIGHTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce ShamAN-Q, a sub-1-bit post-training quantization method that extends NanoQuant by replacing each its diagonal reconstruction geometry with a tractable dense curvature metric, using a general paradigm popularized by the Shampoo optimizer. For each linear weight, ShamAN-Q fits a Kronecker product to the empirical Fisher information matrix of a small calibration set by Kullback–Leibler minimization, forming a Mahalanobis reconstruction loss from the result. The continuous ADMM updates from NanoQuant become solutions to Sylvester equations, while its discrete projection and deployment format remain unchanged. Because the curvature is local to a given set of weights, ShamAN-Q re-measures the input curvature statistic for each layer immediately before layer factorization, periodically refreshing all statistics on the partially quantized model. ShamAN-Q also redistributes the uniform rank from NanoQuant across layers at the same total number of bits. On Qwen3-Base, ShamAN-Q lowers WikiText-2 perplexity at ≈1 bpw from 27.56 to 22.96 (0.6B), 19.21 to 16.72 (1.7B), and 14.29 to 13.80 (4B) while matching or improving zero-shot accuracy on the Eleuther LM Evaluation Harness. On 0.6B, ShamAN-Q at ≈0.8 bpw matches the published perplexity of NanoQuant at ≈1.0 bpw.

1 INTRODUCTION

The energy consumption of language models is an omnipresent, growing concern, affecting all scales from those of frontier data centers down to small edge devices. Awareness of this cost grows alongside demand for data sovereignty, independence from model providers, and fast local intelligence. Beyond the cost of the computation itself, an often overlooked source of language model energy cost is the repeated movement of data. For causal language model inference, processing a prompt in parallel during prefill is often compute-bound. A long prompt produces a large effective batch size B, and assuming $n \times n$ matrices, the resulting matrix–matrix multiplication moves $O(n^2)$ data for $O(Bn^2)$ computation. Decoding, by contrast, is often bandwidth-bound, since each generated token requires transferring the entire active set of parameters from memory to the compute cores, before counting any host–device or device–device transfers required by models that do not fit on a single

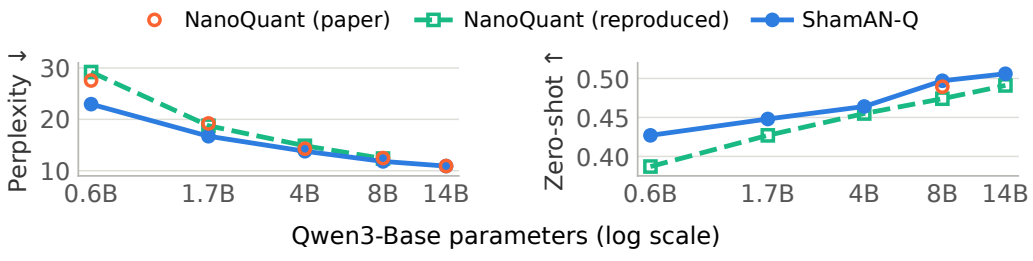

Figure 1: WikiText-2 perplexity (PPL) (left, lower is better) and zero-shot mean on reasoning tasks in Eleuther LM Evaluation Harness (right, higher is better) vs. Qwen3-Base size at a 1.0 bpw budget. ShamAN-Q matches or improves on NanoQuant in both PPL and zero-shot mean at every size. All points are from Table 1; only reported results are plotted.

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.

accelerator. In the extreme, batch size $B = 1$ reduces to matrix–vector multiplication with $O(n^2)$ data movement for only $O(n^2)$ computation.

Shrinking the stored weights therefore directly attacks the dominant cost of decoding, and large language models are increasingly deployed under memory and bandwidth budgets that their weights alone exceed. Post-training quantization (PTQ) to 4 bits per weight (bpw) is now well established (Frantar et al., 2023), and even 2 bpw is maturing (Lin et al., 2024), but the *sub-1-bit* regime requires a structural change, since a scalar codebook cannot spend less than one bit on every weight. NanoQuant (Chong et al., 2026) instead represents each decoder linear layer by two low-rank binary factors and two short FP16 scale vectors (Eq. 1). By doing so, NanoQuant changes the dominant storage from one bit per matrix entry to $r(m + n)$ bits per layer at factor rank $r \gg 16$. At this compression level, nearly every weight is perturbed, and as a result accuracy depends not only on the size of the reconstruction error but also on which input and output directions absorb it.

NanoQuant handles unequal channel scales through diagonal preconditioning, after which its optimization process minimizes a Frobenius reconstruction loss. This diagonal preconditioning acts as a proxy for second-order information specific to each channel, but it does not capture second-order relationships between channels. ShamAN-Q extends NanoQuant by replacing the channel-wise scaling with a dense metric that captures these channel correlations. In this regard, ShamAN-Q is to NanoQuant what Shampoo (Gupta et al., 2018) is to diagonal AdaGrad (Duchi et al., 2011). One-sided Hessian-aware PTQ methods such as GPTQ (Frantar et al., 2023) model input sensitivity while treating all output directions alike. The Fisher Information matrix of a weight matrix captures both, but its $mn \times mn$ size is prohibitive for an $m \times n$ weight. Approximating it with an $n \times n$ input statistic and an $m \times m$ output statistic makes this information tractable (Martens & Grosse, 2015) (§3.2).

ShamAN-Q estimates dense curvature statistics from a small calibration set while limiting the influence of massive-activation tokens. As quantization changes the model, these statistics are regularized and refreshed; they simultaneously serve to guide low-rank reconstruction and rank allocation at NanoQuant's total bit budget, thus preserving its binary representation and deployment format. At this budget, ShamAN-Q lowers WikiText-2 perplexity on Qwen3-Base from 27.56 to 22.96 at 0.6B, 19.21 to 16.72 at 1.7B, and 14.29 to 13.80 at 4B while matching or improving zero-shot accuracy on the Eleuther LM Eval Harness; at 0.8 bpw, ShamAN-Q at 0.6B matches NanoQuant's published 1.0 bpw perplexity (§5). We now describe the primary contributions of this paper:

- A dense two-sided reconstruction metric extension of NanoQuant for sub-1-bit PTQ: KL-Shampoo fit, and a fresh input statistic with periodic full refresh (§4.1, §4.3), resulting in a modified ADMM whose continuous updates are solutions to Sylvester equations. Other components, such as the Sign-Value Independent Decomposition (SVID) projection (Pouransari et al., 2020; Xu et al., 2020), dual updates, scale extraction, and packed format structures remain unchanged (§4.3).
- A redistribution of NanoQuant's uniform rank budget at bit parity from a combination of depth, layer type, and a measurement of reconstruction loss for each layer (§4.2).
- Consistent decreases in perplexity paired with increases in zero-shot reasoning performance over NanoQuant on Qwen3-Base from 0.6B to 14B at identical $\lesssim$ 1-bit budgets, attributed to each component (§5).

2 Related Work

Binary and Sub-1-bit Quantization of LLMs. Weight-only PTQ is already an established paradigm at 4 and 2 bpw (Frantar et al., 2023; Lin et al., 2024). Pushing further, binary methods either train with quantization-aware objectives, as in the cases of BitNet (Wang et al., 2023) and OneBit (Xu et al., 2024), or partially binarize and keep salient weights at higher precision, as in the case of PB-LLM (Shang et al., 2023); Tu et al. (2026), in contrast, revisits 1-bit optimization from pre-trained weights. Generally, in-place binarization with full-precision scales is bounded below by one bit per weight, and grouping metadata pushes the effective rate higher. NanoQuant (Chong et al., 2026) breaks this bound in the PTQ setting by factorizing each weight into two low-rank binary matrices and two scale vectors, initialized by a latent-binary ADMM under diagonal preconditioning and refined block by block (§3.1).

Hessian-aware PTQ and Kronecker Curvature Statistics. GPTQ (Frantar et al., 2023) demonstrates a form of Hessian-aware PTQ by weighting the reconstruction loss with the input second moment, a one-sided Hessian surrogate that models input sensitivity while treating all output directions alike. The second order optimizers K-FAC (Martens & Grosse, 2015) and Shampoo (Gupta et al., 2018) approximate the layer Fisher by a Kronecker product of input and output statistics, and BaKron (Birnick & Saab, 2026) applies similar Kronecker-structured Hessians to weight quantization. Two estimators fit the Kronecker product to the empirical Fisher directly, the nearest Kronecker product (NKP) in Frobenius norm (Mei et al., 2023; Morwani et al., 2025) and the matrix-normal maximum-likelihood fit of KL-Shampoo (Lin et al., 2026; Sun & Wei, 2026). §3.2 presents both, and in ShamAN-Q we adopt the KL fit because of how it weights massive-activation tokens (Sun et al., 2024).

ADMM for Structured Compression. The alternating direction method of multipliers (Boyd et al., 2011) suits objectives that pair a continuous reconstruction loss with a hard discrete constraint. NanoQuant introduces LB-ADMM, which decouples the low-rank reconstruction from the binary structure by using SVID-projected consensus variables. ShamAN-Q retains this effective decoupling and its Euclidean consensus penalties, so that only the continuous subproblems change. These subproblems change from right-sided normal equations (Eq. 5) to two-sided Sylvester equations.

3 Background

3.1 NanoQuant

Representation and bit cost. NanoQuant (Chong et al., 2026) deploys each linear layer, with weight $\mathbf{W} \in \mathbb{R}^{m\times n}$ and factor rank r, in the two-scale form

$$\widehat{\mathbf{W}} = \mathrm{Diag}(\mathbf{s}_1)\,\mathbf{U}_{\pm1}\,\mathbf{V}_{\pm1}^{\top}\,\mathrm{Diag}(\mathbf{s}_2), \qquad \mathbf{U}_{\pm1} \in \{\pm1\}^{m\times r}, \quad \mathbf{V}_{\pm1} \in \{\pm1\}^{n\times r}, \tag{1}$$

using FP16 scale vectors of lengths m and n. Since the signs require $r(m+n)$ bits, and the scales $16(m+n)$ bits, a rank-r layer costs

$$\mathrm{bits}(r) = (r+16)(m+n), \qquad \mathrm{BPW} = (r+16)(m+n)/mn. \tag{2}$$

Pipeline. Phase 1, *global calibration*, runs a calibration set through the full-precision teacher $\mathcal{M}$ and collects, per linear layer and token t, the layer input $\mathbf{x}_t \in \mathbb{R}^n$ and output gradient $\boldsymbol{\delta}_t \in \mathbb{R}^m$. Phase 2, *block reconstruction*, processes decoder blocks in order. Within a block, Step 1 (TuneFP) tunes the remaining full-precision weights to reproduce the teacher's block output from the already-quantized prefix's activations, and is repeated between layer factorizations. Step 2 initializes each linear layer's binary factors and scales by the LB-ADMM below. Step 3 (TuneLatentSTE) jointly tunes the block's continuous latents $\mathcal{U}, \mathcal{V}$ and scales $\mathbf{s}_1, \mathbf{s}_2$ under a weighted MSE block loss with the straight-through estimator (Bengio et al., 2013), after which $\mathbf{U}_{\pm1} = \mathrm{sign}(\mathcal{U})$ and $\mathbf{V}_{\pm1} = \mathrm{sign}(\mathcal{V})$ are packed. Phase 3 (TuneScalesKD), tuning only the scale vectors of every layer, minimizes the forward KL divergence between the next-token distributions of the teacher and quantization models.

Diagonal preconditioning (Step 2-1). NanoQuant performs one measurement of the per-input-feature activation second moments and the per-output-feature output gradient second moments for the full-precision model, after robust token clipping, and shrinks them toward their mean,

$$\mathbf{d}_{\mathrm{in}}^2 = \frac{1}{N}\sum_t \mathbf{x}_t \odot \mathbf{x}_t, \qquad \mathbf{d}_{\mathrm{out}}^2 = \frac{1}{N}\sum_t \boldsymbol{\delta}_t \odot \boldsymbol{\delta}_t, \qquad \mathbf{d}^2 \leftarrow (1-\gamma)\,\mathbf{d}^2 + \gamma\,\mathrm{mean}(\mathbf{d}^2)\,\mathbf{1}, \tag{3}$$

with $\gamma \in [0,1]$. These shrunken moments then produce $\widetilde{\mathbf{D}}_{\mathrm{in}} = \mathrm{Diag}(\mathbf{d}_{\mathrm{in}})$, $\widetilde{\mathbf{D}}_{\mathrm{out}} = \mathrm{Diag}(\mathbf{d}_{\mathrm{out}})$, the transformed target $\widetilde{\mathbf{W}} = \widetilde{\mathbf{D}}_{\mathrm{out}}\mathbf{W}\widetilde{\mathbf{D}}_{\mathrm{in}}$, and the loss $\|\widetilde{\mathbf{E}}\|_F^2 = \mathrm{tr}\left(\mathbf{E}^{\top}\widetilde{\mathbf{D}}_{\mathrm{out}}^2\mathbf{E}\widetilde{\mathbf{D}}_{\mathrm{in}}^2\right)$ where $\mathbf{E} = \mathbf{W} - \widehat{\mathbf{W}}$ and $\widetilde{\mathbf{E}} = \widetilde{\mathbf{D}}_{\mathrm{out}}\mathbf{E}\widetilde{\mathbf{D}}_{\mathrm{in}}$. This loss comprises the two-sided metric of §3.2 with diagonal statistics.

LB-ADMM (Step 2-2). The continuous optimization variables are $\mathbf{U} \in \mathbb{R}^{m\times r}$ and $\mathbf{V} \in \mathbb{R}^{n\times r}$, and they correspond with consensus variables $\mathbf{Z}_U$ and $\mathbf{Z}_V$, which are restricted to the SVID family $\mathcal{C}_{d,r} = \{\mathrm{Diag}(\mathbf{a})\mathbf{S}\,\mathrm{Diag}(\mathbf{b}) : \mathbf{a} \in \mathbb{R}_+^d,\ \mathbf{b} \in \mathbb{R}_+^r,\ \mathbf{S} \in \{\pm1\}^{d\times r}\}$. NanoQuant then solves

$$\min_{\substack{\mathbf{U},\mathbf{V},\mathbf{Z}_U,\mathbf{Z}_V \\ \mathbf{U}=\mathbf{Z}_U,\ \mathbf{V}=\mathbf{Z}_V}} \frac{1}{2}\|\widetilde{\mathbf{W}} - \mathbf{U}\mathbf{V}^{\top}\|_F^2 + \frac{\lambda}{2}\left(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2\right) + \mathcal{I}_{\mathcal{C}_{m,r}}(\mathbf{Z}_U) + \mathcal{I}_{\mathcal{C}_{n,r}}(\mathbf{Z}_V) \tag{4}$$

by a scaled-dual ADMM, with ridge λ and a penalty ρ_k that grows linearly over K iterations. Taking $\alpha_k = \rho_k + \lambda$, with the consensus target $\mathbf{C}_U^k = \mathbf{Z}_U^k - \mathbf{\Lambda}_U^k$ being formed using the scaled dual $\mathbf{\Lambda}_U^k$, the $\mathbf{U}$ update is the right-sided $r \times r$ normal equation

$$\mathbf{U}^{k+1}\big((\mathbf{V}^k)^\top \mathbf{V}^k + \alpha_k \mathbf{I}_r\big) = \widetilde{\mathbf{W}}\mathbf{V}^k + \rho_k \mathbf{C}_U^k, \tag{5}$$

while the $\mathbf{V}$ update is analogous. As shown for $\mathbf{Z}_U$, the consensus variables are projected according to $\mathbf{Z}_U^{k+1} = \mathrm{SVID}(\mathbf{U}^{k+1} + \mathbf{\Lambda}_U^k)$, where SVID takes the sign of its argument and the three largest singular values of its absolute value, and the scaled duals are updated as $\mathbf{\Lambda}_U^{k+1} = \mathbf{\Lambda}_U^k + \mathbf{U}^{k+1} - \mathbf{Z}_U^{k+1}$ (Appendix C).

Scale extraction (Step 2-3). The final proxies $\mathbf{U}^K + \mathbf{\Lambda}_U^K$ and $\mathbf{V}^K + \mathbf{\Lambda}_V^K$ are magnitude-balanced. Their signs are become $\mathbf{U}_{\pm 1}, \mathbf{V}_{\pm 1}$, their row-wise mean absolute values give the scales, and the inverse of the diagonal maps return the scales to the original coordinates (Appendix C).

3.2 Kronecker Curvature Statistics: Shampoo and KL-Shampoo

Empirical Fisher and the two-sided metric. For calibration token t, the weight gradient is the rank-one matrix $\boldsymbol{\delta}_t \mathbf{x}_t^\top$, and the empirical layer Fisher information, in the original coordinates of $\mathrm{vec}(\mathbf{W})$, is given by $\widehat{\mathbf{H}}_{\mathrm{cal}} = \sum_t \mathrm{vec}(\boldsymbol{\delta}_t \mathbf{x}_t^\top)\,\mathrm{vec}(\boldsymbol{\delta}_t \mathbf{x}_t^\top)^\top \in \mathbb{R}^{mn \times mn}$. A Kronecker approximation can estimate this matrix as $\widehat{\mathbf{H}}_{\mathrm{cal}} \approx \mathbf{A} \otimes \mathbf{G}$ using an input-side statistic $\mathbf{A} \in \mathbb{R}^{n \times n}$ and an output-side statistic $\mathbf{G} \in \mathbb{R}^{m \times m}$ (Martens & Grosse, 2015; Gupta et al., 2018). We call these *curvature statistics* and reserve *factors* for the low-rank weight matrices $\mathbf{U}$ and $\mathbf{V}$. Under column-major vectorization, $\mathbf{A}$ and $\mathbf{G}$ act respectively on input and output coordinates, and the metric induced on an error $\mathbf{E}$ is

$$\|\mathbf{E}\|_{\mathbf{G},\mathbf{A}}^2 := \mathrm{tr}\big(\mathbf{E}^\top \mathbf{G} \mathbf{E} \mathbf{A}\big) = \mathrm{vec}(\mathbf{E})^\top (\mathbf{A} \otimes \mathbf{G})\,\mathrm{vec}(\mathbf{E}). \tag{6}$$

All computations use the two statistics directly, so the full $mn \times mn$ Kronecker product never needs to be formed. NanoQuant uses as its loss the diagonal case $\mathbf{A} = \widetilde{\mathbf{D}}_{\mathrm{in}}^2$, $\mathbf{G} = \widetilde{\mathbf{D}}_{\mathrm{out}}^2$, which is equivalent to $\mathbf{A} = \mathbf{I}_n$, $\mathbf{G} = \mathbf{I}_m$ in the transformed coordinates.

Shampoo. Shampoo (Gupta et al., 2018) preconditions a matrix gradient $\nabla_t = \boldsymbol{\delta}_t \mathbf{x}_t^\top$ with two accumulated statistics,

$$\mathbf{G}_{\mathrm{Sh}} = \sum_t \nabla_t \nabla_t^\top = \sum_t \|\mathbf{x}_t\|^2\, \boldsymbol{\delta}_t \boldsymbol{\delta}_t^\top, \qquad \mathbf{A}_{\mathrm{Sh}} = \sum_t \nabla_t^\top \nabla_t = \sum_t \|\boldsymbol{\delta}_t\|^2\, \mathbf{x}_t \mathbf{x}_t^\top, \tag{7}$$

and updates according to $\mathbf{W} \leftarrow \mathbf{W} - \eta\, \mathbf{G}_{\mathrm{Sh}}^{-1/4}\, \nabla\, \mathbf{A}_{\mathrm{Sh}}^{-1/4}$, which corresponds in vectorized form with the preconditioner $(\mathbf{A}_{\mathrm{Sh}}^{1/2} \otimes \mathbf{G}_{\mathrm{Sh}}^{1/2})^{-1/2}$. Morwani et al. (2025) show that the square of Shampoo's approximation is one power-iteration step toward the optimal Kronecker approximation of $\widehat{\mathbf{H}}_{\mathrm{cal}}$, so the exponent $1/4$ per statistic is the inverse square root of an approximate curvature.

Estimators as token-weighted second moments. Kronecker fits of $\widehat{\mathbf{H}}_{\mathrm{cal}}$ all share the form $\mathbf{G} \propto \sum_t w_t^x\, \boldsymbol{\delta}_t \boldsymbol{\delta}_t^\top$, $\mathbf{A} \propto \sum_t w_t^\delta\, \mathbf{x}_t \mathbf{x}_t^\top$, differing only in their token weights. The nearest-Kronecker-product (NKP) fit (Mei et al., 2023; Morwani et al., 2025) minimizes $\|\widehat{\mathbf{H}}_{\mathrm{cal}} - \mathbf{A} \otimes \mathbf{G}\|_F$ over positive semidefinite statistics, with fixed-point weights $w_t^x = \mathbf{x}_t^\top \mathbf{A}\, \mathbf{x}_t$, $w_t^\delta = \boldsymbol{\delta}_t^\top \mathbf{G}\, \boldsymbol{\delta}_t$; Shampoo's statistics coem from its first alternation from $\mathbf{A} = \mathbf{I}_n$, $\mathbf{G} = \mathbf{I}_m$. The KL-Shampoo fit (Lin et al., 2026; Sun & Wei, 2026) is the matrix-normal maximum-likelihood estimate

$$(\mathbf{A}, \mathbf{G}) \in \arg\min_{\mathbf{A},\mathbf{G} \succ \mathbf{0}} D_{\mathrm{KL}}\Big(\mathcal{N}(\mathbf{0}, \widehat{\mathbf{H}}_{\mathrm{cal}}) \,\Big\|\, \mathcal{N}(\mathbf{0}, \mathbf{A} \otimes \mathbf{G})\Big), \qquad \mathrm{tr}(\mathbf{A}) = n, \tag{8}$$

where the trace constraint removes the scale ambiguity $(c\mathbf{A}) \otimes (c^{-1}\mathbf{G}) = \mathbf{A} \otimes \mathbf{G}$. Stationarity then gives the coupled fixed point

$$\mathbf{G} = \frac{1}{n} \sum_t \big(\mathbf{x}_t^\top \mathbf{A}^{-1} \mathbf{x}_t\big)\, \boldsymbol{\delta}_t \boldsymbol{\delta}_t^\top, \qquad \mathbf{A} = \frac{1}{m} \sum_t \big(\boldsymbol{\delta}_t^\top \mathbf{G}^{-1} \boldsymbol{\delta}_t\big)\, \mathbf{x}_t \mathbf{x}_t^\top, \tag{9}$$

so the token weights are the *inverse* quadratic forms. The KL-Shampoo optimizer tracks these statistics by an exponential moving average and updates $\mathbf{W} \leftarrow \mathbf{W} - \eta\, \mathbf{G}^{-1/2}\, \nabla\, \mathbf{A}^{-1/2}$, with exponent $1/2$ per statistic because $\mathbf{A} \otimes \mathbf{G}$ now approximates $\widehat{\mathbf{H}}_{\mathrm{cal}}$ directly.

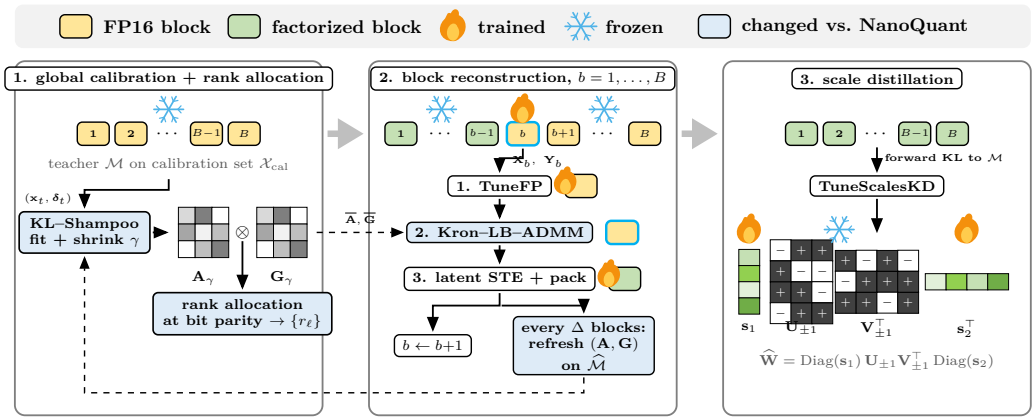

Figure 2: ShamAN-Q end to end. Phase 1 runs the calibration set through the frozen teacher, fits and shrinks the curvature statistics $(\mathbf{A}_\gamma, \mathbf{G}_\gamma)$ of every linear layer, and allocates ranks at bit parity. Phase 2 processes decoder blocks in order. The factorized prefix and the untouched full-precision suffix are frozen while block b is tuned (TUNEFP), factorized layer by layer by Kron–LB–ADMM under the transported statistics $(\overline{\mathbf{A}}, \overline{\mathbf{G}})$ (Figure 3), and refined by latent STE. The statistics of the unfactorized layers are refreshed on the partially quantized model after every Δ blocks. Phase 3 distills only the scale vectors against the teacher, with the sign matrices frozen. Flames mark trained parameters, snowflakes frozen ones, and blue blocks denote differences between ShamAN-Q and NanoQuant.

4 SHAMAN-Q

ShamAN-Q keeps the same representation as NanoQuant and similarly adopts a three-phase pipeline (§3.1), but changes the geometry used to fit the low-rank factors at four interfaces. Curvature is estimated during the initial calibration (§4.1), used during rank allocation (§4.2), applied to the continuous factor updates, and refreshed with respect to the partially quantized model (§4.3). All components related to SVID, scale extraction, latent tuning, packing, and scale distillation are unchanged. Figure 2 shows the three phases and marks their changed stages, while Figure 3 details the curvature-aware ADMM fitting of one layer, which is the primary extension over NanoQuant. The complete pipeline is given in Appendix C.

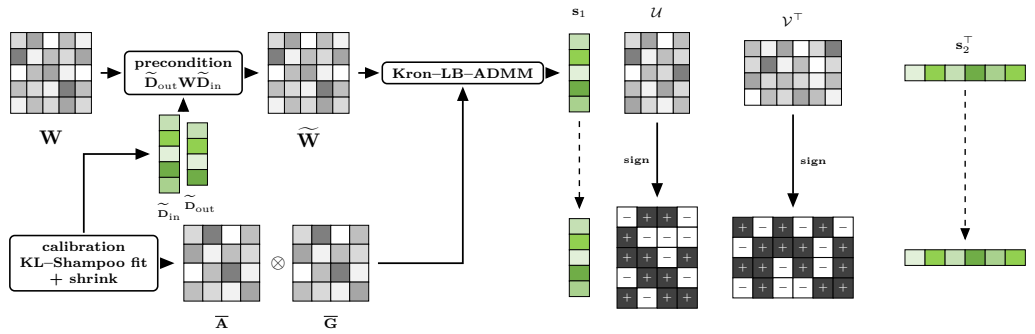

Figure 3: ShamAN-Q pipeline for one layer. Calibration fits and shrinks the curvature statistics, yielding the transported metric $(\overline{\mathbf{A}}, \overline{\mathbf{G}})$ and diagonal maps $(\widetilde{\mathbf{D}}_{\text{in}}, \widetilde{\mathbf{D}}_{\text{out}})$. Preconditioning maps $\mathbf{W}$ to $\widetilde{\mathbf{W}}$. Kron–LB–ADMM returns the continuous weight factors $(\mathcal{U}, \mathcal{V})$ and scale vectors $(\mathbf{s}_1, \mathbf{s}_2)$. The transported statistics alter only the continuous $\mathbf{U}/\mathbf{V}$ optimization component; Euclidean consensus leaves SVID, dual updates, scale extraction, and sign extraction unchanged from NanoQuant.

4.1 Dense Kronecker Curvature from Global Calibration

Curvature estimation. Where NanoQuant keeps only the diagonal of $\widehat{\mathbf{H}}_{\mathrm{cal}}$ (Eq. 3), ShamAN-Q keeps its full Kronecker structure; $(\mathbf{A}, \mathbf{G})$ are fitted for each linear layer by three alternating passes of the KL-Shampoo fixed point (Eq. 9) over the calibration set, with Cholesky inverses. The KL fit is preferred to NKP for its token weighting (§3.2). The NKP weights $\mathbf{x}_t^\top \mathbf{A}\, \mathbf{x}_t$ are quartic in activation magnitude, so the few tokens corresponding with massive activations (Sun et al., 2024) dominate the leading eigenvalues. The KL weights $\mathbf{x}_t^\top \mathbf{A}^{-1} \mathbf{x}_t$, in contrast, are bounded leverage scores, and the resulting statistics have condition numbers of order 40 instead of several hundred. The estimator consumes only the pairs $(\mathbf{x}_t, \boldsymbol{\delta}_t)$ and never materializes $\widehat{\mathbf{H}}_{\mathrm{cal}}$.

Scale-matched shrinkage. The fitted statistics are regularized before any coordinate change,

$$\mathbf{A}_\gamma = (1-\gamma)\mathbf{A} + \gamma\, \tfrac{\mathrm{tr}(\mathbf{A})}{n}\, \mathbf{I}_n, \qquad \mathbf{G}_\gamma = (1-\gamma)\mathbf{G} + \gamma\, \tfrac{\mathrm{tr}(\mathbf{G})}{m}\, \mathbf{I}_m, \qquad \gamma \in [0,1]. \tag{10}$$

Each identity has the same mean eigenvalue as the statistic it regularizes, so shrinkage changes anisotropy without arbitrarily changing its scale: there is no additional damping term. The diagonal of the shrunken statistic is NanoQuant's shrunken $\mathbf{d}^2$ of Eq. 3, and NanoQuant's robust token clipping is kept when the statistics are accumulated. The fit and shrinkage of this section form the calibration stage of Figure 3. They run once at calibration and at each periodic refresh, while the diagonal maps, transport, and tempering of §4.3 run per layer.

4.2 Non-uniform Rank Allocation at Bit Parity

NanoQuant gives every layer the same bits per weight: setting $\mathrm{BPW} = \beta$ in Eq. 2 yields

$$r_\ell^\star = \beta\, m_\ell n_\ell / (m_\ell + n_\ell) - 16, \qquad \beta = 1, \tag{11}$$

giving r_ℓ^{uni} when floored to a multiple of 32. ShamAN-Q keeps the total $N_{\mathrm{bits}} = \sum_\ell \mathrm{bits}_\ell(r_\ell^{\mathrm{uni}})$ (bit parity) and redistributes it across layers in steps of 32 with rank ceiling $\min(m_\ell, n_\ell)$. Given $b_\ell \in \{1, \ldots, B\}$, the block of layer ℓ and its type $t(\ell)$ (q, k, v, o, gate, up, down), both allocators use the depth prior $\omega_\ell = \exp\big(\kappa\big(\frac{b_\ell - 1}{B-1} - \frac{1}{2}\big)\big)\, w_{t(\ell)}$, with $\kappa = 0.6$, so the last block gets more bits than the first by a factor of $e^{0.6} \approx 1.8$. The *hand table* sets budgets $\beta_\ell \propto \beta\, \omega_\ell$—where the type weights w_t are contained in a range from 0.85 (q) to 1.15 (down), taken from the block-loss jump each layer type caused when binarized in the 4B logs—and rounds them to ranks meeting exactly N_{bits}. The *measured sensitivity* allocator instead probes each layer with a short dense-curvature LB-ADMM (§4.3) at three ranks, scores the result by the Gauss–Newton error $J_\ell(r) = \|\mathbf{W}_\ell - \widehat{\mathbf{W}}_\ell(r)\|^2_{\mathbf{G}_{\gamma,\ell}, \mathbf{A}_{\gamma,\ell}}$ using the layer's own metric, fits a power law, multiplies its level by the depth prior with all $w_t = 1$, and hands out 32-rank steps greedily by predicted loss decrease per bit. The probes rank the seven layer types within a block well, but under-value late blocks, which the prior supplies. Appendix B gives the type weights, rounding, and probe protocol.

4.3 Block Reconstruction Pipeline

Blocks are processed in order, and within each block every linear layer is factorized according to NanoQuant's per-layer schedule of §3.1. Step 1 (TuneFP) and Step 3 (TuneLatentSTE, sign extraction, and packing) are unchanged; ShamAN-Q changes only the preconditioning and the continuous solves inside Step 2.

Periodic curvature refresh. The Fisher information is local to the full-precision weights, and the model drifts from that point as blocks are binarized. ShamAN-Q corrects for this at two time scales: a per-layer fresh input statistic inside Step 2-1, and a periodic refresh at block boundaries. For every Δ blocks, the statistics of all *not-yet-factorized* layers are re-estimated with one warm-started pass of Eq. 9 through the current partially quantized model $\widehat{\mathcal{M}}$, so that later blocks see the statistics they will actually receive, and are shrunk with γ as before.

Step 2-1: Dense-Curvature Preconditioning. NanoQuant measures $\widetilde{\mathbf{D}}_{\mathrm{in}}$ once, on the full-precision model (Eq. 3). Under the block-wise schedule, by the time layer ℓ is factorized, its inputs come from an already quantized prefix and from full-precision layers that TuneFP has just modified. Immediately before ADMM for each layer, we therefore re-measure its input second moment on

exactly those inputs, $\mathbf{A}_{\text{fresh}} = \frac{1}{N}\sum_{t=1}^{N} \mathbf{x}_t \mathbf{x}_t^\top$, tkaen over the N calibration tokens with no gradient weighting. Shrunken by Eq. 10, it then takes the place of $\mathbf{A}_\gamma$ for that layer, while the output-side statistic $\mathbf{G}_\gamma$ is that of either the calibration time or most recent refresh.

The diagonal preconditioners are the square roots of the diagonals of the shrunken statistics, floored at $\epsilon > 0$ to keep both maps invertible (Eq. 18), while $\widetilde{\mathbf{W}}$ and $\widetilde{\mathbf{E}}$ are taken as in §3.1. Rather than refitting in these coordinates, we transport the same shrunken statistics by congruence, which preserves the Mahalanobis error exactly:

$$\overline{\mathbf{A}} = \widetilde{\mathbf{D}}_{\text{in}}^{-1} \mathbf{A}_\gamma \widetilde{\mathbf{D}}_{\text{in}}^{-1}, \qquad \overline{\mathbf{G}} = \widetilde{\mathbf{D}}_{\text{out}}^{-1} \mathbf{G}_\gamma \widetilde{\mathbf{D}}_{\text{out}}^{-1}, \qquad \|\mathbf{E}\|_{\mathbf{G}_\gamma, \mathbf{A}_\gamma}^2 = \left\|\widetilde{\mathbf{E}}\right\|_{\overline{\mathbf{G}}, \overline{\mathbf{A}}}^2, \tag{12}$$

which is obtained by substituting $\mathbf{E} = \widetilde{\mathbf{D}}_{\text{out}}^{-1} \widetilde{\mathbf{E}} \widetilde{\mathbf{D}}_{\text{in}}^{-1}$ into Eq. 6 and cycling the trace. Notwithstanding the floor, the transported statistics are unit-diagonal correlation matrices, so the dense metric combines NanoQuant's diagonal preconditioning with $(\overline{\mathbf{G}}, \overline{\mathbf{A}})$. Thus NanoQuant is recovered exactly by $\overline{\mathbf{A}} = \mathbf{I}_n$ and $\overline{\mathbf{G}} = \mathbf{I}_m$.

The one-bit reconstruction error is large ($\|\mathbf{E}\| \approx 0.4$–$0.5\,\|\mathbf{W}\|$), well outside the neighborhood in which the Fisher is an accurate curvature, so before the ADMM solver is applied, each transported statistic is replaced by its trace-preserving power (*spectral tempering*). Writing $\overline{\mathbf{A}} = \mathbf{Q}\,\mathrm{Diag}(\mathbf{c})\mathbf{Q}^\top$, we then have

$$\overline{\mathbf{A}} \;\longleftarrow\; \frac{\sum_i c_i}{\sum_i c_i^p}\, \mathbf{Q}\,\mathrm{Diag}(\mathbf{c}^{\,p})\mathbf{Q}^\top, \qquad p = \tfrac{1}{2}, \tag{13}$$

and likewise for $\overline{\mathbf{G}}$. This halves the log-condition number, and as a result, the data term trusts the dominant curvature directions less. The square root is the regret-optimal metric under an uncertain curvature. Tempering is free given the cached eigendecomposition. It leaves the identity unchanged, and beats both $p = 1$ and Shampoo's $p = \frac{1}{4}$ (§5.4). All uses of $\overline{\mathbf{A}}, \overline{\mathbf{G}}$ below refer to the tempered statistics; their eigensystems and $\mathbf{M} := \overline{\mathbf{G}}\,\widetilde{\mathbf{W}}\,\overline{\mathbf{A}}$ are cached per layer (Algorithm 1 in Appendix C).

Step 2-2: Dense-Curvature LB-ADMM. Relative to NanoQuant's objective (Eq. 4), ShamAN-Q replaces the Frobenius reconstruction term by the transported metric and keeps everything else. In the transformed coordinates it solves

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V},\mathbf{Z}_U,\mathbf{Z}_V} \quad & \frac{1}{2}\left\|\widetilde{\mathbf{W}} - \mathbf{U}\mathbf{V}^\top\right\|_{\overline{\mathbf{G}},\overline{\mathbf{A}}}^2 + \frac{\lambda}{2}\left(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2\right) + \mathcal{I}_{\mathcal{C}_{m,r}}(\mathbf{Z}_U) + \mathcal{I}_{\mathcal{C}_{n,r}}(\mathbf{Z}_V) \\ \text{subject to} \quad & \mathbf{U} = \mathbf{Z}_U, \quad \mathbf{V} = \mathbf{Z}_V. \end{aligned} \tag{14}$$

with $\mathbf{Z}_U$ and $\mathbf{Z}_V$ in the SVID family of §3.1. The reconstruction term is Mahalanobis, but the two augmented consensus penalties are Euclidean. This keeps SVID, dual updates, the ρ_k schedule, and residuals identical to NanoQuant, so only the continuous solves change. Figure 3 follows one layer from calibration through preconditioning, the continuous Kron–LB–ADMM solve, and scale and sign extraction; Algorithm 2 in Appendix C lists the solver in full.

Take $\alpha_k = \rho_k + \lambda$ and let $\mathbf{C}_U^k = \mathbf{Z}_U^k - \mathbf{\Lambda}_U^k$ be the consensus target formed with the scaled dual $\mathbf{\Lambda}_U^k$. Holding $\mathbf{V}^k$ fixed, stationarity of Eq. 14 gives

$$\overline{\mathbf{G}}\,\mathbf{U}^{k+1}\mathbf{R}_V^k + \alpha_k \mathbf{U}^{k+1} = \mathbf{M}\mathbf{V}^k + \rho_k \mathbf{C}_U^k, \qquad \mathbf{R}_V^k = (\mathbf{V}^k)^\top \overline{\mathbf{A}}\,\mathbf{V}^k, \tag{15}$$

and the $\mathbf{V}$ update is obtained analogously, by exchanging the roles of $(\overline{\mathbf{G}}, \mathbf{U})$ and $(\overline{\mathbf{A}}, \mathbf{V})$ and using $\mathbf{M}^\top$ in place of $\mathbf{M}$. These are Mahalanobis Sylvester equations, rather than NanoQuant's right-sided $r \times r$ normal equations (Eq. 5). Because the augmented penalties are Euclidean, the left curvature statistic does not cancel. Both have the form $\mathbf{C}\mathbf{X}\mathbf{R} + \alpha\mathbf{X} = \mathbf{B}$, with $\mathbf{C}$ a tempered, transported curvature statistic and $\mathbf{R} \in \mathbb{R}^{r\times r}$ positive semidefinite. With the curvature eigensystem $\mathbf{C} = \mathbf{Q}\,\mathrm{Diag}(\mathbf{c})\mathbf{Q}^\top$ cached once per layer and $\mathbf{R} = \mathbf{S}\,\mathrm{Diag}(\boldsymbol{\nu})\mathbf{S}^\top$ recomputed for each update, the solution in the joint eigenbasis is

$$\mathbf{X} = \mathbf{Q}\mathbf{Y}\mathbf{S}^\top, \qquad \mathbf{Y}_{i,j} = \left[\mathbf{Q}^\top \mathbf{B}\mathbf{S}\right]_{i,j} / (c_i \nu_j + \alpha). \tag{16}$$

Because $\alpha = \rho + \lambda > 0$, every denominator is positive even for semidefinite statistics. We observe that taking $\overline{\mathbf{G}} = \mathbf{I}_m$ and $\overline{\mathbf{A}} = \mathbf{I}_n$ recovers Eq. 5. One eigendecomposition per statistic per layer comprises the added cost (Appendix D), and the SVID projection and its inner iteration count are independent of $\overline{\mathbf{A}}, \overline{\mathbf{G}}$ (Appendix C).

Table 1: WikiText-2 perplexity and zero-shot mean across a suite of reasoning tasks for <1 bpw Qwen3-Base models at the same total bits as NanoQuant (Chong et al., 2026), with both reported and reproduced values. The upper block uses a 1.0 bpw budget, the middle block a 0.8 bpw budget, and the lower block a 0.55 bpw budget. Unreported results are marked with –.

		PPL ↓					0-shot mean ↑				
bpw	Method	0.6B	1.7B	4B	8B	14B	0.6B	1.7B	4B	8B	14B
≈1.0	NanoQuant (paper)	27.56	19.21	14.29	12.47	10.92	–	–	–	0.4894	–
	NanoQuant (repr.)	29.21	18.76	14.86	12.38	11.49	0.387	0.427	0.455	0.474	0.498
	ShamAN-Q	**22.96**	**16.72**	**13.80**	**11.82**	**10.91**	**0.427**	**0.448**	**0.464**	**0.497**	**0.506**
≈0.8	NanoQuant (paper)	33.79	25.31	19.33	14.83	12.88	–	–	–	–	–
	NanoQuant (repr.)	42.22	22.37	17.99	15.63	–	0.387	0.414	**0.431**	0.443	–
	ShamAN-Q	**27.51**	**18.84**	**17.27**	**14.77**	–	**0.398**	**0.426**	0.430	**0.447**	–
≈0.55	NanoQuant (paper)	52.94	33.74	32.86	20.04	17.06	–	–	–	–	–
	NanoQuant (repr.)	62.65	32.40	39.49	22.49	–	0.355	0.380	**0.409**	**0.419**	–
	ShamAN-Q	**38.80**	**25.63**	**24.24**	**18.21**	–	**0.362**	**0.384**	0.397	0.410	–

Step 2-3: Unchanged Scale Extraction. The magnitude balancing and scale extraction that NanoQuant uses are applied verbatim to the final proxies in the transformed coordinates, followed by its un-preconditioning through the inverse diagonal maps (Appendix C). The resulting sign matrices initialize the latents $\mathcal{U}, \mathcal{V}$ of Step 3, and the block loss, packing, and TUNESCALESKD all come from NanoQuant as well (§3.1).

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Implementation and Environment. ShamAN-Q is implemented on top of the released NanoQuant code and follows its protocol wherever §4 does not change it. The 0.6B to 4B results use a single NVIDIA A100-80GB GPU with fp64 eigendecompositions. The 8B and 14B results use fp32 eigendecompositions on two to four GPUs, which was verified to be lossless at 1.7B. Hyperparameters and the engineering measures that keep the dense recipe within budget are given in Appendix A.

Models and Datasets. We quantize Qwen3-Base at 0.6B, 1.7B, and 4B, 8B, and 14B parameters (Qwen Team, 2025), calibrating on 128 WikiText-2 (Merity et al., 2016) sequences of 2048 tokens (seed 0). At 8B and 14B the curvature statistics and their refreshes use 512 sequences, while block reconstruction and distillation still use 128. We report both WikiText-2 perplexity at context 2048 and the mean zero-shot accuracy from the Eleuther LM Evaluation Harness (Gao et al., 2024). All numbers are single runs.

Baselines. The baseline is NanoQuant at the same total number of bits, presented both as published (Chong et al., 2026) and as our own paper-faithful reproduction. Every ShamAN-Q model spends the *same total number of bits* as NanoQuant. That is, at the 1 bpw budget, 0.973 bpw for 0.6B and 0.986 bpw for 1.7B and 4B, and at 0.8 bpw, 0.773 / 0.783 / 0.793 / 0.794 bpw for 0.6B / 1.7B / 4B / 8B. At 0.55 bpw the actual values are 0.527 / 0.535 / 0.544 / 0.545 bpw, and the 8B curvature statistics use 128 sequences. Only the rank allocator (§4.2) differs across sizes.

Next Token Prediction. Table 1 compares ShamAN-Q with NanoQuant at identical bit budgets. PPL is lower at every model size, and the gap narrows as the model grows. At 0.8 bpw the PPL margin over NanoQuant widens at 1.7B and 4B (13 to 26% and 3 to 11%), narrows to 0.4% at 8B, and at 0.6B and 1.7B ShamAN-Q at 0.8 bpw matches or beats NanoQuant at 1.0 bpw. Figure 1 shows both metrics across model sizes through 14B. The right block reports the zero-shot mean. NanoQuant publishes zero-shot accuracies for Qwen3 only at 8B, and at 4B the rank allocator trades perplexity against zero-shot accuracy (§5.2).

Table 2: Rank allocators at the same bpw. WikiText-2 PPL with zero-shot mean. "Measured" is probed sensitivity × depth prior.

Qwen3-Base	Hand table	Measured
4B	**13.55** / 0.450	13.80 / **0.463**

Table 3: End-to-end wall-clock time for reproduced NanoQuant and ShamAN-Q on Qwen3-Base at 1 bpw on 1× NVIDIA A100 80 GB.

Size	NanoQuant	ShamAN-Q	Overhead
0.6B	1 h 12	1 h 59	65%
1.7B	1 h 18	2 h 15	73%
4B	2 h 30	4 h 30	80%

Table 4: Cumulative attribution at Qwen3-1.7B-Base, uniform ranks, 0.986 bpw, WikiText-2 perplexity.

Configuration	PPL
NanoQuant, published	19.21
NanoQuant, our reproduction	18.76
NKP statistics, untempered	19.21
+ tempering $p = \frac{1}{2}$	17.46
+ fresh input statistic and refresh	17.28
KL-Shampoo statistics, untempered	17.04
+ tempering $p = \frac{1}{2}$ (ShamAN-Q)	**16.72**

5.2 Rank Allocation at Bit Parity

Table 2 compares the allocators of §4.2 at the same total bits. At 0.6B the measured sensitivity × depth prior is best, 2.5 perplexity below uniform ranks. At 4B the measured allocator gives 13.80 against 13.55 from the hand table but restores the zero-shot mean (0.463 vs. 0.450); both are single runs. The depth prior alone thus does not recover what the type weights in the hand table yields at that width, and the two allocators trade perplexity against zero-shot accuracy.

5.3 Quantization Cost

ShamAN-Q uses the same deployment format as NanoQuant, so inference kernels, memory footprint, and decoding throughput are unchanged. The cost is incurred during quantization, as shown in Table 3. Appendix D gives the asymptotic accounting and Appendix A the timing breakdown and cost measures.

5.4 Ablation Studies

Table 4 adds the modifications of §4 cumulatively at 1.7B with uniform ranks. Our reproduction of NanoQuant is already 0.45 below the published number. A dense NKP fit without tempering gives nothing over the diagonal baseline. Tempering the same statistics comprises the single largest step, and replacing the NKP fit by the KL-Shampoo fit improves both the untempered and the tempered models. Only the tempering and rank-allocation steps are individually outside noise. The fresh input statistic and periodic refresh of §4.3 are a consistent gain at every block boundary, though their aggregate effect in Table 4 is within single-run noise. For the tempering exponent, $p = 1$ (none) and $p = 1/4$ (Shampoo) were both worse than $p = 1/2$ at 1.7B.

6 Conclusion

We introduced ShamAN-Q, which replaces the diagonal reconstruction geometry in NanoQuant with a dense metric estimated by a KL-Shampoo-style Kronecker fit of the empirical Fisher Information matrix. We also identify useful processing techniques for robust application of the curvature estimated under uncertainty, including shrinkage, tempering, and periodic refreshes. The continuous ADMM updates of NanoQuant then become Sylvester equation solutions, which can be computed in a cached eigenbasis, while other components of NanoQuant, including crucially the packed format, remain unchanged, so inference can inherit the speed of the customized kernel. At the same total number of bits, ShamAN-Q lowers WikiText-2 PPL and increases zero-shot reasoning performance of $\lesssim 1$ bpw Qwen3-Base models from 0.6B to 14B. Future work includes parallelizing the computation and using further approximations to the dense curvature statistics that yield fast matrix multiplication, both to reduce wall clock time. In addition, as the depth prior and calibration budget were tuned at 0.6B, tuning would likely help, and understanding scaling behavior is left as a promising direction for future investigation.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods. We have not used generative AI tools for proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, or interpreting results. Generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation, cleaning and reformating datasets, and supporting qualitative and thematic data analysis are not applicable to this work.

Additionally, we used generative AI tools for creating scientific figures, suggesting experimental parameters, brainstorming, identifying some relevant literature, and some editing. We have reviewed all AI-assisted work. LLM-generated code was verified via tests reviewed by humans prior to implementation. AI-generated figures were produced as TikZ that was manually revised by humans, and AI-generated writing was manually reviewed by humans. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work is intended to reduce the energy consumption required to run AI models, which has been a globally increasing concern due to scarcity of resources, limited production capacity, and environmental impact. On the surface, we might expect that reducing the energy cost of model inference would have a positive impact on these issues. However, this should be considered in the context of Jevon's paradox, where a decreasing energy cost for using an item results in increased aggregate usage due to more widespread access and adoption to the point that total energy consumption due to the item increases. This effect, while not guaranteed, remains a possibility here.

REPRODUCIBILITY STATEMENT

To aid with reproducibility, we provide source code with configuration files to run experiments found in the paper, which we intend to open source as part of the publication. The implementation details of the algorithms and relevant hyperparameters are contained within §4 and appendices. In addition, in the appendices we have included some discussion of directions that were unsuccessful in our explorations.

REFERENCES

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Johann Birnick and Rayan Saab. Bakron: Efficient quantization with kronecker-factored hessians. *arXiv preprint arXiv:2608.06291*, 2026.

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

Hyochan Chong, Dongkyu Kim, Changdong Kim, and Minseop Choi. NanoQuant: Efficient Sub-1-Bit Quantization of Large Language Models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=qiZDlnvWTR.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL https://jmlr.org/papers/v12/duchi11a.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.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, 2018.

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 LLM compression and acceleration. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

Wu Lin, Scott C. Lowe, Felix Dangel, Runa Eschenhagen, Zikun Xu, and Roger Grosse. Understanding and improving Shampoo and SOAP via Kullback-Leibler Minimization. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 92799–92826, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/95a704bd2fdf8ef8242b4adcc7ce3c93-Paper-Conference.pdf.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, 2015.

Jonathan Mei, Alexander Moreno, and Luke Walters. Kradagrad: kronecker approximation-domination gradient preconditioned stochastic optimization. In *Uncertainty in Artificial Intelligence*, pp. 1412–1422. PMLR, 2023.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Depen Morwani, Itai Shapira, Nikhil Vyas, Eran Malach, Sham Kakade, and Lucas Janson. A new perspective on shampoo's preconditioner. *International Conference on Learning Representations*, 2025:5802–5822, May 2025.

Hadi Pouransari, Zhucheng Tu, and Oncel Tuzel. Least squares binary quantization of neural networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2986–2996. IEEE, 2020.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. PB-LLM: Partially binarized large language models. *arXiv preprint arXiv:2310.00034*, 2023.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *First Conference on Language Modeling (COLM)*, 2024. arXiv:2402.17762.

Ruotong Sun and Ermin Wei. Pro-KLShampoo: Projected KL-Shampoo with Whitening Recovered by Orthogonalization. *arXiv preprint arXiv:2605.06316*, 2026.

Zhijun Tu, Jian Li, Yuanyuan Xi, Siqi Liu, Chuanjian Liu, Hanting Chen, Jie Hu, and Yunhe Wang. Rethinking 1-bit optimization leveraging pre-trained large language models. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=xdS18gI4e8.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. BitNet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

Junhao Xu, Xie Chen, Shoukang Hu, Jianwei Yu, Xunying Liu, and Helen Meng. Low-bit quantization of recurrent neural network language models using alternating direction methods of multipliers. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7939–7943. IEEE, 2020.

Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. OneBit: Towards extremely low-bit large language models. In *Advances in Neural Information Processing Systems*, 2024.

A Implementation Details

Hyperparameters. All runs follow the protocol used by NanoQuant wherever §4 does not change it: 128 WikiText-2 calibration sequences of 2048 tokens selected with seed 0 (512 for the curvature statistics at 8B and 14B), shrinkage $\gamma = 0.2$, NanoQuant's robust token clipping, $K = 400$ LB-ADMM iterations with the linear ρ_k schedule, two FP16 scale vectors per layer, and TuneFP/TuneLatentSTE/TuneScalesKD with learning rates $10^{-4}/10^{-5}/10^{-6}$, batch sizes $4/1/1$, 8 epochs each (at 8B and 14B the first two stages use 2, 4, 6, and 8 epochs for the q and k, v and o, gate, and up and down layers) with cosine decay, NanoQuant's weighted MSE block loss, and forward-KL scale-only distillation. ShamAN-Q adds three KL-Shampoo passes at calibration, the tempering exponent $p = \frac{1}{2}$, the refresh period $\Delta = 7$ for the 28-block 0.6B and 1.7B models and $\Delta = 9$ for the 36-block 4B model and 8B models, and $\Delta = 10$ for the 40-block 14B model, the fresh input statistic, and the rank allocator listed in Table 1. The measured allocator probes each layer at $\{0.5, 1, 1.5\} \times r_\ell^\star$ at 0.6B and 1.7B and at $\{0.6, 1, 1.4\} \times r_\ell^\star$ from 4B upward, with $K = 50$. Every $n \times n$ and $m \times m$ eigendecomposition in the reported runs is computed in fp64.

Keeping the dense recipe within budget. The recipe costs more than NanoQuant per block: dense statistics per layer, one eigendecomposition per statistic per ADMM call, the fresh input measurement, the refresh passes, the rank probe, and late blocks whose allocated ranks reach 2560 at 4B (§5.3). With allocated ranks at 4B, ADMM accounts for 260 s of a 473 s block at block 7, and for about 40 % of the block with uniform ranks. Four measures keep this within budget. (i) The fp32 statistics of all layers do not fit next to the model at 4B (about 36 GB), so the calibration passes accumulate the Kronecker statistics on the GPU over layer groups sized to a memory budget instead of streaming to the CPU, reducing 4B calibration from about 5 h to about 35 min. The KL fit's per-pass inverses are Cholesky solves on the accumulation device, reducing 1.7B calibration from 55 to 9 min. (ii) The fresh input statistic of §4.3 is measured once per shared-input group and its eigendecomposition cached for the group's ADMM calls, saving three of seven measurements and three large eigendecompositions per block. (iii) The $n \times n$ eigendecompositions can run in fp32: the eigenvalues are clamped and the result feeds a sign projection, and 1.7B perplexity was unchanged (16.66 vs. 16.72 with fp64). The reported 1.7B and 4B results used fp64, so this is a cost measure only. (iv) Calibration statistics, the rank probe, every per-layer ADMM solution (content-addressed by weight bytes, curvature tensors, and ADMM settings), per-block checkpoints, and the pre-distillation model are cached, so an interrupted run resumes from its last block in minutes. Refresh costs are small, as demonstrated by how three refreshes cost 4 min in total at 1.7B. The rank probe cost is given in Appendix B.

B Rank Allocation Details

This appendix completes §4.2. Both allocators start from the uniform ranks r_ℓ^{uni} of Eq. 11, keep their total N_{bits}, and move ranks in steps of 32 with ceiling $\min(m_\ell, n_\ell)$.

Hand table. The per-layer budget is $\beta_\ell = \beta\, \zeta\, \omega_\ell$ with the depth prior ω_ℓ of §4.2 and type weights $w = (\text{q } 0.85,\ \text{k } 0.90,\ \text{v } 1.10,\ \text{o } 1.00,\ \text{gate } 1.00,\ \text{up } 1.05,\ \text{down } 1.15)$, set from the median relative block-loss jump each layer type caused when binarized in the 4B logs, and ζ chosen so that $\sum_\ell \beta_\ell m_\ell n_\ell = \beta \sum_\ell m_\ell n_\ell$. Ranks are obtained from β_ℓ by Eq. 11, floored to multiples of 32, and then moved in ± 32 steps, largest relative deviation from β_ℓ first, until the total equals N_{bits}.

Measured sensitivity times depth prior. After calibration, every layer is solved by a short run of the dense-curvature LB-ADMM of §4.3 ($K = 50$) at the three candidate ranks $\{0.5, 1, 1.5\} \times r_\ell^\star$ at 0.6B and 1.7B and $\{0.6, 1, 1.4\} \times r_\ell^\star$ from 4B upward, and the deployed matrix is scored by the Gauss–Newton weight error

$$J_\ell(r) = \left\| \mathbf{W}_\ell - \widehat{\mathbf{W}}_\ell(r) \right\|^2_{\mathbf{G}_{\gamma,\ell}, \mathbf{A}_{\gamma,\ell}}, \tag{17}$$

with the layer's own shrunken statistics before the trace normalization, so that values are comparable across layers. A power law $J_\ell(r) = e^{a_\ell} r^{-\eta_\ell}$ is fitted through the three probes and its level multiplied by the depth prior of §4.2 with all $w_t = 1$. Starting from rank 32 everywhere, 32-rank steps are handed to the layer with the largest predicted loss decrease per added bit, $\big(J_\ell(r) - J_\ell(r+32)\big)/\big(32(m_\ell + n_\ell)\big)$, until N_{bits} is reached. The measured curves rank the seven layer types within a block well (Spearman 0.8–0.96 against logged block-loss jumps) but under-value late blocks, which is what the prior supplies. The probe shares each statistic's eigendecomposition across its three candidate ranks and is cached, taking 707 s at 0.6B.

C ALGORITHMS

This appendix lists the procedures of §4.3 and the unchanged NanoQuant components they call (§3.1), written in the transformed coordinates.

Diagonal maps and transformed target (Step 2-1). With the maximum applied elementwise, the diagonal preconditioners are

$$\mathbf{d}_{\text{in}} = \sqrt{\max\{\text{diag}(\mathbf{A}_\gamma), \epsilon \mathbf{1}_n\}}, \qquad \widetilde{\mathbf{D}}_{\text{in}} = \text{Diag}(\mathbf{d}_{\text{in}}),$$
$$\mathbf{d}_{\text{out}} = \sqrt{\max\{\text{diag}(\mathbf{G}_\gamma), \epsilon \mathbf{1}_m\}}, \qquad \widetilde{\mathbf{D}}_{\text{out}} = \text{Diag}(\mathbf{d}_{\text{out}}), \tag{18}$$

and the transformed target, transformed error, and cached metric target are

$$\widetilde{\mathbf{W}} = \widetilde{\mathbf{D}}_{\text{out}} \mathbf{W} \widetilde{\mathbf{D}}_{\text{in}}, \qquad \widetilde{\mathbf{E}} = \widetilde{\mathbf{D}}_{\text{out}} \mathbf{E} \widetilde{\mathbf{D}}_{\text{in}}, \qquad \mathbf{M} := \overline{\mathbf{G}}\, \widetilde{\mathbf{W}}\, \overline{\mathbf{A}}, \tag{19}$$

where $\mathbf{M}$ uses the tempered transported statistics of Eq. 13 and is cached together with their eigensystems. Algorithm 1 lists the step.

Algorithm 1 PREPAREMETRICTARGET for one layer (Step 2-1).

Require: $\mathbf{W}$, shrunken statistics $(\mathbf{A}_\gamma, \mathbf{G}_\gamma)$, with $\mathbf{A}_\gamma$ from the fresh measurement of §4.3, floor ϵ, tempering exponent p

1: Form $(\widetilde{\mathbf{D}}_{\text{in}}, \widetilde{\mathbf{D}}_{\text{out}})$ by Eq. 18
2: $\widetilde{\mathbf{W}} \leftarrow \widetilde{\mathbf{D}}_{\text{out}} \mathbf{W} \widetilde{\mathbf{D}}_{\text{in}}$
3: Transport to $(\overline{\mathbf{A}}, \overline{\mathbf{G}})$ by Eq. 12, do not refit
4: Eigendecompose $\overline{\mathbf{A}}, \overline{\mathbf{G}}$ and temper both by Eq. 13
5: $\mathbf{M} \leftarrow \overline{\mathbf{G}} \widetilde{\mathbf{W}} \overline{\mathbf{A}}$, cache the tempered eigensystems
6: **return** $\widetilde{\mathbf{W}}$, both diagonal maps, and the tempered metric state

Continuous block updates (Step 2-2). Set $\alpha_k = \rho_k + \lambda$. Holding $\mathbf{V}^k$ fixed, define

$$\mathbf{R}_V^k = (\mathbf{V}^k)^\top \overline{\mathbf{A}} \mathbf{V}^k, \qquad \mathbf{B}_U^k = \mathbf{M}\mathbf{V}^k + \rho_k \mathbf{C}_U^k, \tag{20}$$

so that Eq. 15 reads $\overline{\mathbf{G}}\, \mathbf{U}^{k+1} \mathbf{R}_V^k + \alpha_k \mathbf{U}^{k+1} = \mathbf{B}_U^k$. After this update, define

$$\mathbf{R}_U^{k+1} = (\mathbf{U}^{k+1})^\top \overline{\mathbf{G}} \mathbf{U}^{k+1}, \qquad \mathbf{B}_V^k = \mathbf{M}^\top \mathbf{U}^{k+1} + \rho_k \mathbf{C}_V^k, \tag{21}$$

which yields

$$\overline{\mathbf{A}}\, \mathbf{V}^{k+1} \mathbf{R}_U^{k+1} + \alpha_k \mathbf{V}^{k+1} = \mathbf{B}_V^k. \tag{22}$$

Both systems are solved by Eq. 16, and these two solves are denoted by KRONUSOLVE and KRONVSOLVE in Algorithm 2.

Euclidean proxy projection. For a proxy $\mathbf{P} \in \mathbb{R}^{d \times r}$, SVID computes

$$\mathbf{S} = \text{sign}(\mathbf{P}), \qquad (\mathbf{a}, \sigma, \mathbf{b}) = \text{TOPSINGULARTRIPLET}(|\mathbf{P}|), \tag{23}$$

with nonnegative unit vectors $\mathbf{a}, \mathbf{b}$ and zeros in $\mathbf{P}$ mapped to $+1$ by sign. Defining $\mathbf{p} = \sqrt{\sigma}\mathbf{a}$ and $\mathbf{q} = \sqrt{\sigma}\mathbf{b}$, the projected proxy is

$$\text{SVID}(\mathbf{P}) = (\sigma \mathbf{a}\mathbf{b}^\top) \odot \mathbf{S} = \text{Diag}(\mathbf{p}) \mathbf{S}\, \text{Diag}(\mathbf{q}). \tag{24}$$

The triplet is obtained by the same power iteration as NanoQuant. Since the consensus penalties in Eq. 14 are Euclidean, neither Eq. 24 nor its inner iteration count depends on $\overline{\mathbf{A}}$ or $\overline{\mathbf{G}}$.

Scale extraction (Step 2-3). The final proxies $\mathbf{P}_U, \mathbf{P}_V$ live in the transformed coordinates of Eq. 19. NanoQuant's magnitude balancing (its Eq. 7) and scale extraction (its Eq. 8) are applied to them verbatim, yielding sign matrices $\mathbf{S}_U, \mathbf{S}_V$ and transformed scale vectors $\widetilde{\mathbf{s}}_1, \widetilde{\mathbf{s}}_2$. The return to the original weight coordinates is

$$\mathbf{s}_1 = \mathbf{d}_{\text{out}}^{-1} \odot \widetilde{\mathbf{s}}_1, \qquad \mathbf{s}_2 = \mathbf{d}_{\text{in}}^{-1} \odot \widetilde{\mathbf{s}}_2, \tag{25}$$

which is exactly NanoQuant's un-preconditioning step with the diagonal maps of Eq. 18. FINALIZE in Algorithm 2 denotes these operations.

Algorithm 2 KRON-LB-ADMM for one transformed layer (Step 2-2).

Require: Prepared state from Algorithm 1, rank r, iterations K, penalty schedule ρ_k, ridge λ

Initialize $\mathbf{U}^0, \mathbf{V}^0$ as in NanoQuant
$\mathbf{Z}_U^0 \leftarrow \text{SVID}(\mathbf{U}^0), \quad \mathbf{Z}_V^0 \leftarrow \text{SVID}(\mathbf{V}^0)$
Initialize scaled duals $\mathbf{\Lambda}_U^0, \mathbf{\Lambda}_V^0$
for $k = 0, \ldots, K-1$ **do**
  $\mathbf{C}_U^k \leftarrow \mathbf{Z}_U^k - \mathbf{\Lambda}_U^k, \quad \mathbf{C}_V^k \leftarrow \mathbf{Z}_V^k - \mathbf{\Lambda}_V^k$
  $\mathbf{U}^{k+1} \leftarrow \text{KRONUSOLVE}(\mathbf{V}^k, \mathbf{C}_U^k; \overline{\mathbf{G}}, \overline{\mathbf{A}}, \mathbf{M}, \rho_k, \lambda)$
  $\mathbf{V}^{k+1} \leftarrow \text{KRONVSOLVE}(\mathbf{U}^{k+1}, \mathbf{C}_V^k; \overline{\mathbf{G}}, \overline{\mathbf{A}}, \mathbf{M}, \rho_k, \lambda)$
  $\mathbf{Z}_U^{k+1} \leftarrow \text{SVID}(\mathbf{U}^{k+1} + \mathbf{\Lambda}_U^k)$
  $\mathbf{Z}_V^{k+1} \leftarrow \text{SVID}(\mathbf{V}^{k+1} + \mathbf{\Lambda}_V^k)$
  $\mathbf{\Lambda}_U^{k+1} \leftarrow \mathbf{\Lambda}_U^k + \mathbf{U}^{k+1} - \mathbf{Z}_U^{k+1}$
  $\mathbf{\Lambda}_V^{k+1} \leftarrow \mathbf{\Lambda}_V^k + \mathbf{V}^{k+1} - \mathbf{Z}_V^{k+1}$, update Euclidean residuals
end for
$\mathbf{P}_U \leftarrow \mathbf{U}^K + \mathbf{\Lambda}_U^K, \quad \mathbf{P}_V \leftarrow \mathbf{V}^K + \mathbf{\Lambda}_V^K$
return $\text{FINALIZE}(\mathbf{P}_U, \mathbf{P}_V, \widetilde{\mathbf{D}}_{\text{out}}, \widetilde{\mathbf{D}}_{\text{in}})$

End-to-end pipeline. Algorithm 3 shows the complete pipeline. Calibration statistics are first collected from the full-precision teacher, so every initial KL-Shampoo fit is defined in the original layer coordinates, and ranks are allocated once. After TUNEFP and the fresh measurement, $\widetilde{\mathbf{W}}$, the transported and tempered statistics, and $\mathbf{M}$ are recomputed for the current target. Latent tuning and final knowledge distillation act on $\mathcal{U}, \mathcal{V}, \mathbf{s}_1, \mathbf{s}_2$ exactly as in NanoQuant.

Algorithm 3 Dense-curvature ShamAN-Q.

Require: Full-precision teacher $\mathcal{M}$, calibration set $\mathcal{X}_{\text{cal}}$, bit budget β and allocator (§4.2), $\gamma, \epsilon, p, \Delta$, ADMM parameters K, ρ, λ, tuning budgets $T_{\text{pre}}, T_{\text{post}}, T_{\text{glob}}$

// Phase 1: global calibration and rank allocation
Collect $(\mathbf{x}_t, \boldsymbol{\delta}_t)$ for every linear layer on $\mathcal{X}_{\text{cal}}$ through $\mathcal{M}$
for each linear layer ℓ **do**
  $(\mathbf{A}, \mathbf{G})^{(\ell)} \leftarrow$ KL-SHAMPOO by Eq. 9, three passes, shrink by Eq. 10
end for
Allocate ranks $\{r_\ell\}$ at bit parity (§4.2), probing with $K = 50$ if the measured allocator is used
// Phase 2: block reconstruction
$\widehat{\mathcal{M}} \leftarrow \mathcal{M}$
for blocks $b = 1, \ldots, B$ **do**
  if $b > 1$ and $(b-1) \bmod \Delta = 0$ **then**
    Refresh $(\mathbf{A}, \mathbf{G})^{(\ell)}$ of all unfactorized layers with one warm-started pass of Eq. 9 through $\widehat{\mathcal{M}}$, shrink
  end if
  $\mathbf{X}_b \leftarrow \widehat{\mathcal{M}}_{<b}(\mathcal{X}_{\text{cal}}), \quad \mathbf{Y}_b \leftarrow \mathcal{B}_b^{\text{FP}}(\mathbf{X}_b)$
  TUNEFP$(b, \mathbf{X}_b, \mathbf{Y}_b; T_{\text{pre}})$ *// Step 1*
  for shared-input groups $g \in b$ (q k v, o, gate up, down) **do**
    Measure $\mathbf{A}_{\text{fresh}}^{(g)}$ (§4.3) on the current inputs of g, shrink by Eq. 10
    for linear layers $\ell \in g$ with weight $\mathbf{W}^{(\ell)}$ **do**
      PREPAREMETRICTARGET (Algorithm 1) with $\mathbf{A}_\gamma \leftarrow \mathbf{A}_{\text{fresh},\gamma}^{(g)}$ and the current $\mathbf{G}_\gamma^{(\ell)}$ *// Step 2-1*
      $(\mathcal{U}, \mathcal{V}, \mathbf{s}_1, \mathbf{s}_2)^{(\ell)} \leftarrow$ KRON-LB-ADMM at rank r_ℓ by Algorithm 2 *// Steps 2-2, 2-3*
    end for
  end for
  TUNELATENTSTE$(b, \mathbf{X}_b, \mathbf{Y}_b; T_{\text{post}})$ over $\mathcal{U}, \mathcal{V}, \mathbf{s}_1, \mathbf{s}_2$ *// Step 3*
  for linear layers $\ell \in b$ **do**
    $\mathbf{U}_{\pm 1}^{(\ell)} \leftarrow \text{sign}(\mathcal{U}^{(\ell)}), \quad \mathbf{V}_{\pm 1}^{(\ell)} \leftarrow \text{sign}(\mathcal{V}^{(\ell)})$
    PACKBINARY$(\mathbf{U}_{\pm 1}^{(\ell)}, \mathbf{V}_{\pm 1}^{(\ell)})$
  end for
end for
// Phase 3: model reconstruction
TUNESCALESKD$(\widehat{\mathcal{M}}, \mathcal{M}, \mathcal{X}_{\text{cal}}; T_{\text{glob}})$ over $\mathbf{s}_1, \mathbf{s}_2$
return $\widehat{\mathcal{M}}$

D COMPLEXITY ANALYSIS

Table 5 separates calibration, preparation, and one outer ADMM sweep for a layer with N calibration tokens. The dense setup includes forming $\mathbf{M}$ and eigendecomposing the two transported statistics.

Table 5: Per-layer costs for an $m \times n$ weight, factor rank r, and N calibration tokens.

Component	NanoQuant diagonal geometry	ShamAN-Q dense geometry
Curvature state	$\Theta(m+n)$	$\Theta(m^2+n^2)$ statistics and eigensystems
Calibration fit	$\Theta(N(m+n))$	$\Theta(N(m^2+n^2)+m^3+n^3)$ per pass, three passes
Fresh input statistic	–	$\Theta(Nn^2)$ per shared-input group
Refresh	–	one fit pass over unfactorized layers every Δ blocks
Rank probe	–	three $K{=}50$ ADMM solves per layer, once
Setup	$\Theta(mn)$	$\Theta(m^3+n^3+m^2n+mn^2)$
One outer sweep	$\Theta(mnr+(m+n)r^2+r^3)$	$\Theta(mnr+(m^2+n^2)r+(m+n)r^2+r^3)$
SVID inner work	$\Theta(K_{\text{in}}(m+n)r)$	unchanged

Each layer performs two continuous solves per outer iteration and two SVID projections at initialization plus two per iteration. For representation $x \in \{\text{diagonal}, \text{dense}\}$, its reconstruction runtime can be decomposed as

$$T_{x,\ell} = S_{x,\ell} + K_{\text{out},\ell} C^{\text{outer}}_{x,\ell} + (K_{\text{out},\ell}+1) C^{\text{SVID}}_{\ell} + N_{\text{diag},\ell} D_{x,\ell}, \tag{26}$$

where S is setup, C^{outer} is one pair of continuous solves, C^{SVID} is one pair of proxy projections, and D covers diagnostic evaluations. This accounting makes the one-time dense setup explicit rather than hiding it in the iteration cost. In practice, the $(m^2+n^2)r$ term does not dominate: at 4B, where allocated ranks reach $r \approx 1500$–2500 and each layer performs 800 $r \times r$ eigendecompositions over $K = 400$ iterations, removing every $n^2 r$ product changes per-layer ADMM time by only 5–8 %, so the per-iteration $r \times r$ eigendecomposition dominates instead (Appendix E).

E Design Alternatives Not Adopted

The following variants were evaluated and are not part of ShamAN-Q.

Variants within noise. A dense Mahalanobis *block* loss for TuneFP and TuneLatentSTE (NanoQuant's weighted MSE is kept), a spike-plus-flat projection of the curvature statistics, distilling the latent binaries in the KD stage, feature distillation, trainable norm weights, and best-epoch selection in the KD stage, a logit-level objective for the last blocks, lifting the rank ceiling, and spending the rounding remainder to reach exactly 1.0 bpw were all within single-run noise. An explicit per-rank middle scale vector between the two sign matrices (§4.3), carried from ADMM through tuning and KD, was 3.5–5 perplexity *worse* at 0.6B: a mean-one rebalanced export of the same vector was within noise, and a sweep of its learning rate was flat to negative. On the 0.6B four-block screen used for tuning-budget changes (block-3 perplexity 14.15 for the control, single-run noise about 0.2), a plateau stop that ends an epoch loop once the block loss improves by less than 10^{-2} was within noise (14.23), while running one TuneFP round per shared-input group (four rounds per block instead of seven) cost $+0.28$, outside noise, so every round matters, including those between layers that read the same activation.

Speed variants. An inexact Sylvester solve (a stale eigenbasis as preconditioner with Rayleigh, QR, and conjugate-gradient rungs) saved about 10 % of ADMM time at 1.7B and nothing at 4B, and early stopping of ADMM never fires: at iteration 400 every layer still flips between 30 and 20,000 signs per iteration under the linear ρ_k schedule. A low-rank-plus-identity Sylvester step for spectrally projected statistics, which keeps the k_{s} largest and k_{d} smallest eigenvalues of each transported statistic exactly and replaces the rest by their geometric mean so that each update costs $O(nkr + nr^2)$ instead of $O(n^2 r)$ with $k = k_{\text{s}} + k_{\text{d}}$, saved only 5–8 % of 4B ADMM time, because the $r \times r$ eigendecomposition and not the $n^2 r$ products dominates at those widths (Appendix D): the projection itself cost $+0.10$ to $+0.34$ perplexity at block 3 with $k = 64$ per side and was within noise at $k = 256$. It is not used.