THINQUANT: SCALABLE ROTATION LEARNING FOR WEIGHT AND ACTIVATION QUANTIZATION OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learned rotations play an important role in enabling low-bit weight and activation quantization of large language models by smoothing outliers in the activation distribution. State-of-the-art approaches include gradient-based procedures such as SpinQuant and computationally friendlier gradient-free approaches such as DartQuant, but both remain hard to scale to the largest architectures. To address the computational bottlenecks in gradient-free rotation learning, we introduce two ideas for efficiency, (i) a data selection procedure which reduces the required number of calibration data points, and (ii) an exact reduction of the associated optimization on this reduced calibration set. Our data selection procedure exploits the geometric structure of the convex hull of the activations. Using this idea, we show that a carefully selected calibration set with several orders of magnitude fewer activations than state-of-the-art rotation-based methods can match their performance in low-bit quantization settings. Under this extreme data efficiency, the selected activations span an r-dimensional subspace with $r < d$, making optimization over a $d \times d$ rotation equivalent to optimizing a $d \times r$ matrix on the Stiefel manifold. We solve this reduced problem using an efficient ADMM algorithm that iteratively employs thin matrix updates at every step, hence the name ThinQuant. For Llama-3-70B with W4A4KV4 quantization, ThinQuant completes the entire rotation calibration in under 12 minutes and achieves a WikiText-2 perplexity of 5.63, compared with 7.55 for DartQuant, which requires 111 minutes. Unlike SpinQuant and DartQuant, ThinQuant also scales to Llama-3.1-405B on a single H200 GPU, completing rotation calibration in just over 2 hours and achieving WikiText-2 perplexity of 2.97 at W4A4, compared with 3.48 for GPTAQ+QuaRoT.

1 INTRODUCTION

State-of-the-art large language models (LLMs) contain billions of parameters and process long token sequences, leading to large memory footprints and slow inference time. As a result, post-training quantization (PTQ), where a pretrained checkpoint is compressed without retraining from scratch, has become a central tool for making LLM inference more efficient (Dettmers et al., 2022). Weight-only PTQ (e.g., INT8 or INT4 weights with higher-precision activations) substantially reduces memory and storage, and methods such as GPTQ (Frantar et al., 2023), GPTAQ (Li et al., 2025), and ADMM-Q (Lucas et al., 2026) achieve near-lossless accuracy even at 4-bit weight precision. However, weight-only schemes leave the dominant computation—matrix–vector products during inference—executed in higher precision (e.g. FP16 or BF16). Joint weight *and* activation quantization gives greater end-to-end speedups, since the core matrix multiplications can be executed by INT8 or INT4 tensor cores, which offer higher peak throughput than FP16 on modern GPUs.

Joint weight and activation quantization is substantially harder than weight-only quantization. Weights are fixed after training and can be compressed offline using expensive optimization procedures. The resulting quantized tensors are then stored and reused at every inference step. Activations, by contrast, arrive at runtime, are different for every input, and must be quantized on-the-fly with a cheap deterministic rule, typically round-to-nearest (RTN) on a uniform grid. Moreover, it has been observed that for activations a small subset of input channels, often fewer than 1% of dimensions, carry values that are orders of magnitude larger than the remaining channels (Dettmers et al., 2022; Xiao et al., 2023).

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.

Rotations address this by changing the coordinate system of the network before quantization which does not change the full precision outputs, but makes the activations substantially easier to quantize. For an orthogonal transformation $R \in \mathbb{R}^{d\times d}$, a linear map can be equivalently written as $Wx = (WR^\mathsf{T})(Rx)$, so that activations are represented in the rotated basis while the inverse transformation is absorbed into the surrounding weights. Modern LLM quantization methods exploit this invariance at both *global* and *local* scales (Liu et al., 2025; Shao et al., 2025). A global change of basis uses a shared rotation across the hidden representation of multiple layers. Local changes of basis instead act within individual layers or modules, for example inside attention or feed-forward blocks, and can therefore use different rotations at different locations in the network. In both cases, the transformations can be absorbed into adjacent linear operators so that the full-precision network remains functionally identical, while its weights and activations can have substantially more favorable distributions for low-bit quantization. Moreover, rotations can be incorporated into quantized inference pipelines to accelerate inference runtime; we refer the reader to (Liu et al., 2025), Section 4.5.

SpinQuant learns rotations by optimizing an end-to-end loss through the quantized network using a straight-through estimator (Liu et al., 2025). While effective, this requires repeated forward and backward passes through the *full* network, making optimization expensive and potentially prone to overfitting to the fine-tuning objective. DartQuant substantially reduces this cost by replacing end-to-end training with a calibration objective on the activations (Shao et al., 2025). However, this still requires optimizing against a very large collection of activations, which leads to large computational cost. In its standard calibration setting, DartQuant uses a sample α of the tokens generated by $N = 128 \times 2048 \approx 2.6 \times 10^5$ samples from each layer. Across L transformer blocks, the rotation is therefore calibrated using αNL activation vectors, where α is the sampling ratio, set to 10% by default. For Llama-3-70B, $L = 80$, this is on the order of millions of activation vectors. ThinQuant, using geometry-aware activation selection, takes this to extreme efficiency: using only a few hundred activation vectors from the entire network, ThinQuant gives dramatic savings in disk storage ($\approx 2000\times$) and runtime ($\approx 10\times$) for Llama-3-70B with performance comparable to more expensive state-of-the-art methods (e.g. SpinQuant, DartQuant), see Figure 1 and Table 4.

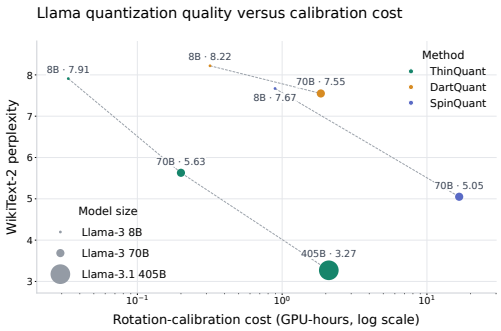


Figure 1: Comparison of Wikitext-2 perplexity versus calibration cost for different rotation-based methods on LLaMA-3 series models at W4A4KV4.

Using our smart activation selection procedure, ThinQuant can operate with a very small calibration dataset, mitigating the shortcomings of existing gradient-free methods such as DartQuant. We briefly provide some intuition for our selection procedure (see Section 3.1 for details). We mitigate the quantization error caused by large outliers in the activations space by controlling the ℓ_∞ norm of the activations after rotation, similar to prior work such as DartQuant and QuaRot (Ashkboos et al., 2024). Our key observation is that these worst-case magnitudes are determined by the *convex hull* of the activations, rather than by the full activation distribution. For a calibration set $\mathcal{A} = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$, the extreme points of the symmetric convex hull $K = \mathrm{conv}(\mathcal{A} \cup (-\mathcal{A}))$ captures the largest absolute projection in every direction. Its extreme points therefore suffice to determine the worst-case objective, while the interior activations can be discarded without changing it. Our data selection procedure thus selects geometrically prominent extreme points in the activations across many possible bases, rather than only those that are outliers in the original coordinates. In practice, this geometric selection lets us replace hundreds of thousands of calibration tokens per layer with a tiny subset of activations.

Given this reduced number of data points, we show that loss functions commonly used to control quantization error (e.g. ℓ_∞, kurtosis) admit an exact reduction to a smaller optimization problem. Generally, the budget B we use for activation vectors (order 100) is much smaller than the hidden dimension d (order thousands). Let $U \in \mathbb{R}^{d\times B}$ collect the selected activation vectors, and let $r = \mathrm{rank}(U) \le B$. Writing a thin factorization $U = QS$, the columns of $Q \in \mathbb{R}^{d\times r}$ are orthonormal, so $Q \in \mathrm{St}(d, r) = \{Y \in \mathbb{R}^{d\times r} : Y^\mathsf{T}Y = I_r\}$. Since the calibration objective depends on the full rotation $R \in \mathbb{R}^{d\times d}$ only through $RU = (RQ)S$, it is determined entirely by the $d \times r$ matrix RQ. We show that optimization over the full orthogonal group is therefore exactly equivalent to optimization over the much smaller Stiefel manifold $\mathrm{St}(d, r)$. Hence ThinQuant, with a few hundred selected

activations, replaces a dense $d \times d$ rotation by a $d \times r$ variable with $r \ll d$, reducing the effective optimization dimension from quadratic to linear in d, with no additional approximation beyond the initial activation selection. We summarize our **contributions** as follows:

- **Reducing the calibration data.** We develop a geometric activation-selection procedure to select activations that are frequently exposed as extreme along random Gaussian directions. This reduces the calibration set required for rotation learning by several orders of magnitude, lowering activation storage and the cost of objective evaluations. We support this selection approach with a dimension-free bound on the expected discrepancy between the full and reduced calibration objectives under Haar-random rotations (Theorem 1).
- **Reducing the optimization dimension.** We show that this reduced calibration set leads to an exact reduction of the rotation-learning problem. For selected activations spanning an r-dimensional subspace with $r < d$, optimizing a $d \times d$ rotation is equivalent to optimizing a $d \times r$ matrix on the Stiefel manifold, with no additional approximation beyond activation selection (Theorem 2). We exploit this equivalence through an ADMM algorithm with thin matrix factorizations, reducing optimization memory and per-iteration cost without requiring forward or backward passes through the model during optimization.
- **Efficient rotation learning at scale.** Combining these reductions, ThinQuant achieves highly competitive perplexity and downstream accuracy on Llama-2 and Llama-3 under W4A4KV4 quantization at substantially lower calibration cost. On Llama-3-70B, ThinQuant reduces total rotation-calibration time from 111 to approximately 12 minutes relative to DartQuant, while improving WikiText-2 perplexity from 7.55 to 5.63 (Table 1). Moreover, to demonstrate the extreme scalability of our method, we employ ThinQuant on Llama-3.1-405B. It runs in just over 2 hours on a single H200 GPU, and achieves WikiText-2 perplexity at W4A4 (2.97 vs 3.48 of GPTAQ+QuaRoT), where other optimization-based rotation learning methods fail to scale under similar hardware constraints.

2 Related work

Post-training quantization. Weight-only quantization reduces parameter storage and memory traffic while retaining higher-precision activations. GPTQ (Frantar et al., 2023) uses approximate second-order information for weight quantization. GPTAQ (Li et al., 2025) extends GPTQ with a different calibration scheme that uses extra closed-form weight updates to compensate for discrepancies between full-precision and quantized layer inputs. AWQ uses activation statistics to protect salient weight channels (Lin et al., 2024), and ADMM-Q uses an ADMM-based method for weight-only quantization (Lucas et al., 2026). Beyond weight-only methods, joint weight and activation quantization additionally enables low-precision matrix multiplications, but must accommodate large outlier values in activation distributions (Dettmers et al., 2022; Xiao et al., 2023). SmoothQuant addresses this difficulty through function-preserving rescaling, transferring quantization difficulty from activations to weights (Xiao et al., 2023). OmniQuant further learns clipping parameters and equivalent transformations through block-wise reconstruction (Shao et al., 2024). Such equivalent transformations change the representation presented to the quantizer without changing the full-precision network output. Our method also falls within this post-training setting, targeting joint low-bit weight and activation quantization. Specifically, we focus on function-preserving rotations, and discuss rotation-based methods below.

Rotation-based quantization. Rotation-based methods provide another way to obtain representations that are more amenable to low-bit quantization. QuIP uses random orthogonal preprocessing to improve weight and Hessian incoherence for weight-only quantization (Chee et al., 2023). QuaRot applies fixed randomized Hadamard rotations to support weight, activation, and KV-cache quantization (Ashkboos et al., 2024), and is generally the fastest method since it is calibration-free. SpinQuant improves substantially on the quality of QuaRot by instead learning rotations by optimizing an end-to-end quantized-network loss, using Cayley updates to maintain orthogonality (Liu et al., 2025), but can be slow since it requires full backward passes through the network. DartQuant reduces this cost by replacing end-to-end optimization with a direct activation-distribution objective, combining an exponential loss with a QR decomposition at every step (Shao et al., 2025). Our work builds on these function-preserving rotation frameworks and focuses on reducing the data and optimization dimensions required for calibration of the rotation.

3 METHOD

Let $X_\ell \in \mathbb{R}^{T\times d}$ contain the calibration activations associated with layer ℓ, with row t written as $x_{\ell,t}^\mathsf{T}$. We adopt the function-preserving rotation placements of prior work (Ashkboos et al., 2024; Liu et al., 2025), which include a global change of basis R_1 and a local change of basis $R_{2,\ell}$ for each layer. The rotation $R_1 \in \mathrm{SO}(d)$ defines a single basis shared across all layers. Namely at every layer X_ℓ is represented as $\widetilde{X}_\ell = X_\ell R_1^\mathsf{T}$ for $\ell = 1,\dots,L$, with the change of basis inverted only at the output as $X_L = \widetilde{X}_L R_1$. In contrast, each $R_{2,\ell}$ acts only locally within layer ℓ, with $\widetilde{X}_\ell = X_\ell R_{2,\ell}^\mathsf{T}$ and $X_\ell = \widetilde{X}_\ell R_{2,\ell}$. In both cases, the rotations can be absorbed into the surrounding weight matrices, including the final inverse of R_1 into the output projection, preserving the full-precision network function without introducing additional online matrix multiplications. Learning these rotations nevertheless leads to large-scale calibration problems. For R_1, a single rotation must be calibrated jointly against activations from every layer, so the amount of calibration data grows with the entire network. For $R_{2,\ell}$, the calibration data are local to a single layer, but the rotation itself is still high-dimensional and a separate optimization problem must be solved for every $\ell = 1,\dots,L$.

ThinQuant addresses these costs with two complementary reductions. First, for learning both R_1 and $R_{2,\ell}$, we select a small set of informative activations (Section 3.1), reducing both calibration storage and the amount of data processed at every optimization step. Second, we exploit the low-dimensional span of these activations to reduce the size of the optimization problem (Section 3.2). Instead of updating a dense $d \times d$ rotation, we optimize a $d \times r$ matrix, where $r \ll d$ is the rank of the selected activation matrix. This reformulation is exact for the selected set and replaces expensive square-matrix orthogonality updates with much cheaper thin matrix factorizations. Our ADMM solver (Section 3.3) operates on this smaller representation to reduce optimization memory and per-iteration cost.

3.1 SELECTING A REDUCED CALIBRATION SET

Rotation calibration. We now turn to the rotation calibration problem. We remark that collecting and storing the full activation matrices can be prohibitively expensive for large models. For example, DartQuant (Shao et al., 2025) explicitly stores a sample of massive calibration activations, requiring approximately 960 GiB of disk storage for LLaMA-3 70B in their default setting used in our experiments (Table 4 in the appendix). Our goal is therefore to select a small subset of activations that approximately preserves the rotation objective (1). For R_1, we consider activations from all layers, $\mathcal{A}_1 = \bigcup_{\ell=1}^{L}\{x_{\ell,t}^{(1)} : 1 \le t \le T\}$, whereas for $R_{2,\ell}$ we use only the corresponding layer, $\mathcal{A}_{2,\ell} = \{x_{\ell,t}^{(2)} : 1 \le t \le T\}$. Apart from this choice of calibration set, the method is applied identically to R_1 and $R_{2,\ell}$. We therefore write $\mathcal{A} = \{x_1,\dots,x_N\} \subset \mathbb{R}^d$ for either collection, $X = [x_1,\dots,x_N] \in \mathbb{R}^{d\times N}$ for its activation matrix, and $R \in \mathrm{SO}(d)$ for the corresponding rotation. Thus $N = LT$ for R_1 and $N = T$ for each $R_{2,\ell}$. For either rotation, a natural worst-case objective including all calibration data is:

$$\min_{R\in\mathrm{SO}(d)} F(R), \qquad F(R) = \|RX\|_\infty = \max_{x\in\mathcal{A}} \|Rx\|_\infty, \tag{1}$$

where $\|\cdot\|_\infty$ denotes the largest absolute entry. Geometrically, the goal is to find an orthonormal basis in which no calibration activation develops a large coordinate. Uniform quantization is governed by the dynamic range of the activation, so the largest coordinate directly leads to a coarser quantization resolution. The ℓ_∞ norm therefore provides a natural worst-case surrogate for suppressing activation outliers and reducing quantization error (Wei et al., 2023; Xiao et al., 2023; Chee et al., 2023; Ashkboos et al., 2024). Our goal is to solve (1) without requiring the full calibration data. Specifically, given an activation budget $B \ll N$, we want a subset $\mathcal{I} \subseteq [N]$ with $|\mathcal{I}| = B$ such that:

$$F_\mathcal{I}(R) = \max_{i\in\mathcal{I}} \|Rx_i\|_\infty \approx F(R). \tag{2}$$

The subset $\mathcal{I}$ is selected before rotation optimization, so it must represent this surrogate as the basis R changes, rather than only in the original coordinate system.

A geometric view of calibration data selection. We first give some intuition on how the calibration data $\mathcal{A}$ in (1) might naturally be reduced. First, notice that the worst-case objective in (1) is determined, for any fixed rotation R, by activations attaining $\max_{x\in\mathcal{A}} |r_k^\mathsf{T} x|$ for one of the rows r_k^T of R. However,

since the rows of R change during optimization (and the optimal R is not known in advance), the maximizer can also change, so activations that are not maximizers in the original basis may become so after rotation. Thus, rather than retaining only activations that are large in the original basis, we want a small subset that retains the activations that may become maximizers during rotation optimization.

Classical convex geometry provides a precise language for describing these extreme directions (Ziegler, 1995). Define the symmetric convex hull $K = \mathrm{conv}\big(\mathcal{A} \cup (-\mathcal{A})\big)$ and its support function $h_K(v) = \max_{z \in K} v^\mathsf{T} z = \max_{x \in \mathcal{A}} |v^\mathsf{T} x|$, defined for every $v \in \mathbb{R}^d$. Since $\mathcal{A}$ is finite and K is a polytope, maximizing the linear functional $x \mapsto v^\mathsf{T} x$ over K attains its optimum at an extreme point of K for every v (Bertsimas and Tsitsiklis, 1997). The candidate maximizers (in x) are therefore confined to $\mathrm{ext}(K) \subseteq \mathcal{A} \cup (-\mathcal{A})$, where $\mathrm{ext}(\cdot)$ is the set of extreme points of the polytope. This holds independently of v, so a single finite set of signed activations suffices to evaluate h_K in every direction. Writing the rows of a rotation R as $r_1^\mathsf{T}, \ldots, r_d^\mathsf{T}$, we can thus write (1) equivalently as:

$$\min_{R \in \mathrm{SO}(d)} F(R) = \max_{1 \le k \le d} h_K(r_k), \tag{3}$$

The key observation here is that changing R changes the viewing direction, but not the underlying polytope K. Figure 2 illustrates this. The directions r_1, r_2, r_3 expose different activations x_1, x_2, x_3, respectively, but each support value can only be attained at an extreme point of the same convex hull. Thus, the relevant boundary point can change as the rotation changes, but activations on the interior of K can never obtain an extreme point. This explains why the full calibration set can be redundant. An activation which is not an extreme point cannot become extreme along any direction. Any subset $\mathcal{A}_\mathcal{I} \subseteq \mathcal{A}$ satisfying $\mathrm{conv}\big(\mathcal{A}_\mathcal{I} \cup (-\mathcal{A}_\mathcal{I})\big) = K$ preserves $F(R)$ for every rotation R. This also connects rotation calibration to the literature on geometric coresets and, in particular, ε-kernels, where small subsets are constructed to approximately preserve the width of the convex hull in all directions (Agarwal et al., 2004).

Selection algorithm. Although the relevant calibration information is concentrated on extreme points, recovering all extreme points is unattractive in our setting. For general polyhedra, deciding if a partial vertex list omits another vertex is NP-complete (Khachiyan et al., 2008). Even for a finite number of points, exact identification of all extreme points would require solving a sequence of high-dimensional convex-hull membership problems. Similarly, ε-kernels, which approximate the directional width of a point set simultaneously over all directions, have worst-case size $\Theta\big(\varepsilon^{-(d-1)/2}\big)$ for preserving directional widths within a relative $(1-\varepsilon)$ factor (Agarwal et al., 2004). With hundreds of thousands of activation vectors in dimensions of several thousand, these are considerably more expensive than the lightweight selection procedure we want.

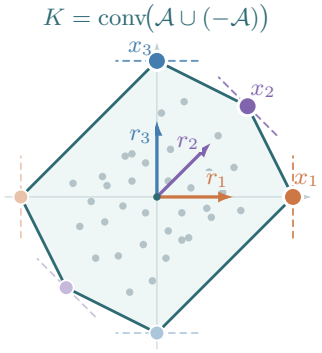


Figure 2: For any direction r, the support value $h_K(r) = \max_{x \in \mathcal{A}} |r^\mathsf{T} x|$ is attained at an extreme point of K. Hence, as the rotation changes, only the exposed boundary activations can determine $F(R)$ and interior activations do not affect the worst-case objective.

A natural and more lightweight approach is to query the support function $h_K(v)$ from (3) along random directions. This idea has precedent in randomized extreme-point detection. In particular, Damle and Sun (2017) generate Gaussian linear functionals, record the points attaining their extrema, and use selection frequency to identify geometrically prominent extreme points. We apply the same random-projection idea to the symmetric hull $K = \mathrm{conv}(\mathcal{A} \cup (-\mathcal{A}))$. We give the procedure in Algorithm 1. We compare the results from this procedure with ℓ_∞-based selection in Table 3 in the appendix, where we retain the B activations with the largest ℓ_∞ norm either in the original basis or after an initial Hadamard rotation.

Algorithm 1 Activation selection

Require: Activations $\mathcal{A} = \{x_1, \ldots, x_N\}$, number of random directions M, total activation budget B

1: $g_1, \ldots, g_M \overset{\text{iid}}{\sim} \mathcal{N}(0, I_d)$ // Gaussian directions
2: $i_j \leftarrow \arg\max_{1 \le i \le N} |g_j^\mathsf{T} x_i|, \quad j = 1, \ldots, M$ // Extreme point?
3: $e_i \leftarrow \sum_{j=1}^{M} \mathbf{1}\{i_j = i\}, \quad i = 1, \ldots, N$ // How often extreme?
4: $\mathcal{I} \leftarrow \mathrm{Top}_B(e)$ // Global ranking
5: **return** $\{x_i : i \in \mathcal{I}\}$

Interpretation of the algorithm. We now give an interpretation of Algorithm 1 in terms of the calibration objective. Notice that (3) describes the calibration objective under the best possible change of basis. Rather than requiring a small subset to preserve this objective for every possible rotation (which is clearly intractable), consider how well it is preserved on average over $R \sim \mathrm{Haar}(\mathrm{SO}(d))$, i.e., for a randomly sampled Haar rotation:

$$\mathbb{E}_{R\sim\mathrm{Haar}(\mathrm{SO}(d))}\left[F(R)\right] = \mathbb{E}_{R\sim\mathrm{Haar}(\mathrm{SO}(d))}\left[\max_{1\le k\le d} h_K(r_k)\right]. \tag{4}$$

Algorithm 1 can be interpreted as selecting a small subset of the activations with which to approximate the objective in (4). For the selected subset $\mathcal{I} \subseteq [N]$, let $K_{\mathcal{I}} = \mathrm{conv}\{\pm x_i : i \in \mathcal{I}\}$ and write $F_{\mathcal{I}}(R) = \max_{1\le k\le d} h_{K_{\mathcal{I}}}(r_k)$ as the corresponding reduced calibration objective. Since $K_{\mathcal{I}} \subseteq K$, we have $F_{\mathcal{I}}(R) \le F(R)$ for every rotation R. Thus the subset objective is $\mathbb{E}_R[F_{\mathcal{I}}(R)]$, and we can measure the quality of the selected calibration set via $\mathbb{E}_R[F(R) - F_{\mathcal{I}}(R)]$, which measures how well the selected activations represent the full calibration objective in (1) under a random change of basis.

Theorem 1 gives a bound on this discrepancy. Specifically, let $q(\mathcal{I})$ denote the probability that, along a random Gaussian direction, the maximum support value is attained by an activation that is not contained in the selected set $\mathcal{I}$. The theorem shows that this probability directly controls the expected difference between the full and reduced calibration objectives. Thus, when Algorithm 1 selects activations that attain the directional maximum for most random Gaussian directions, the reduced objective closely approximates (1) on average over random changes of basis, without requiring recovery of the entire convex hull.

Theorem 1 (Expected Approximation Guarantee)**.** *Let $\mathcal{A} = \{x_1, \dots, x_N\} \subset \mathbb{R}^d$, with $d \ge 2$, and identify activations that coincide up to sign. For any fixed nonempty subset $\mathcal{I} \subseteq [N]$ returned by Algorithm 1, define:*

$$q(\mathcal{I}) = \Pr_{g\sim\mathcal{N}(0,I_d)}\left[\arg\max_{i\in[N]} |g^\mathsf{T} x_i| \notin \mathcal{I}\right],$$

Then, for $R \sim \mathrm{Haar}(\mathrm{SO}(d))$:

$$\mathbb{E}_R[F(R) - F_{\mathcal{I}}(R)] \le \left(\max_{i\in[N]} \|x_i\|_2\right)\sqrt{3q(\mathcal{I})}. \tag{5}$$

The proof is given in Appendix A. The main idea is to relate the Gaussian directions used by Algorithm 1 to the coordinate directions generated by a random rotation. If $R \sim \mathrm{Haar}(\mathrm{SO}(d))$, then each row direction r_k is marginally uniform on the unit sphere. Likewise, for $g \sim \mathcal{N}(0, I_d)$, the normalized vector $g/\|g\|_2$ is uniform on the unit sphere. Thus, projecting an activation onto a normalized Gaussian direction is distributionally equivalent to an individual coordinate after a Haar rotation. Although the rows of R are dependent through orthogonality, the proof bounds the objective gap by a sum of coordinate errors, whose expectation depends only on these marginal distributions.

An appealing feature of Theorem 1 is that the quantity controlling the bound is directly estimated by Algorithm 1. Indeed, for the selected set $\mathcal{I}$, the counts e_i gives the empirical estimate:

$$\widehat{q}_M(\mathcal{I}) = 1 - \frac{1}{M}\sum_{i\in\mathcal{I}} e_i,$$

which is precisely the fraction of sampled Gaussian directions whose maximizing activation is not contained in $\mathcal{I}$. As $M \to \infty$, these empirical frequencies converge to their population probabilities, and hence $\widehat{q}_M(\mathcal{I}) \to q(\mathcal{I})$. We report $\widehat{q}_M(\mathcal{I})$ for both Llama-3.1-405B and Llama-3-8B at $M = 8192$. We see that the counts are highly concentrated. Selecting a single activation already reduces the empirical miss probability $\widehat{q}_M(\mathcal{I})$ to 0.78 and 0.12, respectively, and it reaches zero on both models at $B = 1280$ activations. For Llama-3-8B and Llama-3.1-405B, this is already far less than 0.1% of $\mathcal{A}$. Finally, given the selected index set $\mathcal{I}$ returned by Algorithm 1, we collect the selected activations as columns of the reduced calibration matrix:

$$U = X_{:,\mathcal{I}} = [x_i]_{i\in\mathcal{I}} \in \mathbb{R}^{d\times B}. \tag{6}$$

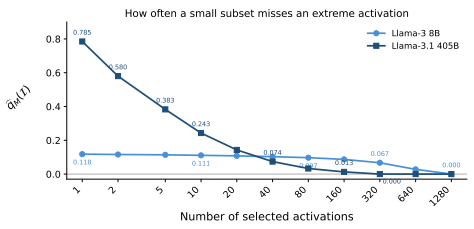


Figure 3: Empirical estimate $\widehat{q}_M(\mathcal{I})$ of the miss probability $q(\mathcal{I})$ in Algorithm 1 as a function of the activation budget B.

We replace the calibration matrix X with the selected matrix $U = [x_i]_{i\in\mathcal{I}} \in \mathbb{R}^{d\times B}$, where $|\mathcal{I}| = B$:

$$\min_{R\in\mathrm{SO}(d)} F_U(R), \qquad F_U(R) = \|RU\|_\infty = \max_{i\in\mathcal{I}} \|Rx_i\|_\infty \tag{7}$$

The calibration procedure thus retains activations that determine the maximum across many directions, while reducing the number of calibration vectors from N to B. This also restricts the span of the retained activations to dimension at most B, which we discuss next.

3.2 An exact reduction in the rotation optimization problem

Once U is fixed, the objective in (7) evaluates R only on the selected activations, all of which lie in $\mathrm{span}(U)$. Any two rotations that agree on this subspace therefore produce the same matrix RU and attain the same loss, regardless of how they act on its orthogonal complement. Thus, rather than tracking the action of R on all of $\mathbb{R}^d$, it suffices to track its action on a basis for $\mathrm{span}(U)$. To make this clear, take an exact QR decomposition $U = Q_U S$, where $r = \mathrm{rank}(U) \le B$. Here, $Q_U \in \mathbb{R}^{d\times r}$ has orthonormal columns spanning $\mathrm{span}(U)$, and $S \in \mathbb{R}^{r\times B}$ contains the coordinates of the selected activations in this basis. The matrix $V = RQ_U$ records the images of these basis vectors under R, and hence determines every rotated activation through $RU = VS$. Moreover, orthogonality of R implies $V^\mathsf{T}V = Q_U^\mathsf{T}R^\mathsf{T}RQ_U = I_r$, so $V \in \mathrm{St}(d, r) := \{V \in \mathbb{R}^{d\times r} : V^\mathsf{T}V = I_r\}$. The following theorem proves the converse when $r < d$, showing that every $V \in \mathrm{St}(d, r)$ can be written as RQ_U for some $R \in \mathrm{SO}(d)$, so optimizing over V is equivalent to optimizing over the full rotation R.

Theorem 2 (Exact selected-subspace reduction)**.** *Let $U = Q_U S \in \mathbb{R}^{d\times B}$ have rank $1 \le r < d$, where $Q_U \in \mathbb{R}^{d\times r}$, $S \in \mathbb{R}^{r\times B}$, and $Q_U^\mathsf{T}Q_U = I_r$. Then $\{RU : R \in \mathrm{SO}(d)\} = \{VS : V \in \mathrm{St}(d, r)\}$. Then for every continuous function $\ell : \mathbb{R}^{d\times B} \to \mathbb{R}$:*

$$\min_{R\in\mathrm{SO}(d)} \ell(RU) = \min_{V\in\mathrm{St}(d,r)} \ell(VS). \tag{8}$$

Every feasible reduced solution lifts to a full rotation with identical objective value.

Proof. The forward inclusion follows from $V = RQ_U$. Conversely, extend Q_U and V to orthogonal bases $[Q_U\ Q_\perp]$ and $[V\ V_\perp]$ and set $R = [V\ V_\perp][Q_U\ Q_\perp]^\mathsf{T}$. Then $RQ_U = V$. Because $r < d$, a sign change in one complement column can enforce $\det R = 1$ without changing RQ_U. Hence $RU = VS$. Since $\mathrm{SO}(d)$ and $\mathrm{St}(d, r)$ are compact and ℓ is continuous, both minima are attained; equality of the feasible image sets therefore implies (8). □

Theorem 2 applies directly to the entrywise ℓ_∞ objective in (7), since $\ell(RU) = \|RU\|_\infty$ is continuous. It also applies without modification to any continuous surrogate objective.

Calibration loss. In general, directly optimizing the ℓ_∞ norm is not attractive, since it requires taking subgradients and can make convergence slow. For optimization, we thus use a fourth-power penalty as a smooth surrogate for the entrywise ℓ_∞ objective. Since $\|RU\|_\infty^4 \le \|RU\|_4^4$, this provides a smooth upper bound on the fourth power of the maximum entry. This choice is motivated by related kurtosis-based approaches to activation outlier suppression (Akhondzadeh et al., 2025). Moreover, we center each rotated activation within the loss because a common shift of all coordinates changes their absolute magnitudes but leaves their range unchanged. This is relevant to the asymmetric activation quantization used in our experiments, where the quantization grid need not be centered at zero. We therefore penalize deviations from each rotated activation's coordinate mean. Specifically, let $C_d = I_d - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\mathsf{T}$, where $\mathbf{1}_d \in \mathbb{R}^d$ is the all-ones vector. We therefore replace the objective in (7) with a surrogate in (9). Since the centered loss is still continuous, Theorem 2 gives:

$$\min_{R\in\mathrm{SO}(d)} \|C_dRU\|_4^4 = \min_{V\in\mathrm{St}(d,r)} \|C_dVS\|_4^4. \tag{9}$$

3.3 Fast optimization and rotation recovery

The reduced problem in (9) replaces the full rotation with a thin matrix V, but the loss remains coupled to the orthogonality constraint through C_dVS. We use the Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011) to separate these components. Specifically, introducing $W = VS$ and $Z = C_dW$, we can write the problem as:

$$\min_{V,W,Z} \|Z\|_4^4 \qquad \text{subject to} \quad W = VS, \quad Z = C_dW, \quad V^\mathsf{T}V = I_r. \tag{10}$$

ADMM alternates between updates of Z, W, and V – each of which can be computed via closed-form updates (see Appendix B) – followed by dual updates that track violations of the two linear constraints. The Z-update handles the fourth-power penalty through independent proximal updates by solving a cubic polynomial per entry. The W-update balances agreement with VS and the centered variable Z through a closed-form least-squares step. Since C_d is a projector, this step requires only column means rather than a dense matrix inverse. Finally, the V-update enforces orthogonality by solving an orthogonal Procrustes problem (Golub and Van Loan, 2013) using a thin $d \times r$ Singular Value Decomposition (SVD). This update remains exact even when S is not orthogonal, because $\|VS\|_F^2 = \|S\|_F^2$ on the Stiefel manifold. All iterations operate on the reduced calibration matrices, so no transformer forward or backward pass is needed during optimization.

Cost. Factoring U costs $O(dB^2)$ once for $B \leq d$. Each iteration costs $O(drB + dr^2)$ and stores $O(dB + dr + rB)$ numbers. When $r = B \ll d$, these become $O(dB^2)$ work and $O(dB)$ storage, replacing a square orthogonality step with $O(d^3)$ work and $O(d^2)$ rotation state.

After optimization, we recover R using the completion in the proof of Theorem 2, with a random Hadamard rotation for the orthogonal complement. Completion leaves $RU = VS$ unchanged, although different completions can act differently on unseen activations outside $\mathrm{span}(U)$.

4 Experiments

4.1 Experimental setup

Models and baselines. We evaluate ThinQuant on LLaMA-2 (7B, 13B, and 70B), LLaMA-3 (8B and 70B) (Grattafiori et al., 2024). All models are evaluated under W4A4(KV4) quantization, with weights, activations, and (when applicable) the KV cache quantized to 4 bits. We use symmetric quantization for weights and asymmetric quantization for activations and the KV cache. As in SpinQuant and DartQuant, we use fixed random Hadamard rotations for R_3 and R_4 (which are KV cache specific); these rotations are not optimized and are enabled for every ThinQuant, DartQuant, and SpinQuant evaluation. Unless otherwise stated, weight quantization is performed using GPTQ throughout all experiments, and is applied after rotation calibration so that GPTQ operates on the rotated weight matrices. We compare against DartQuant and SpinQuant under the same quantization settings. All experiments for all methods are conducted on a single NVIDIA H200 GPU, with the exception of SpinQuant where we use 4 H200 GPUs and report total GPU hours.

Evaluation. We measure language-modeling performance using perplexity (↓) on WikiText-2, C4, and Penn Treebank (PTB). For downstream evaluation, we report the average zero-shot accuracy (↑) across nine tasks using the LM Evaluation Harness (Gao et al., 2023): PIQA (Bisk et al., 2020), ARC-Easy and ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), RTE (Wang et al., 2018), OpenBookQA (Mihaylov et al., 2018), BoolQ (Clark et al., 2019), and Social IQa (Sap et al., 2019).

Calibration. Unless otherwise stated, ThinQuant uses $M = 8192$ independent Gaussian directions in Algorithm 1. For the shared residual rotation R_1, we set the total activation budget to $B = 2L + 1$, where L denotes the number of transformer layers. The rotations are applied at the down-projection and output-projection layers, so this budget is sufficient to admit one activation vector from each projection in every layer, though we select the top B globally as in Algorithm 1. For R_2, we set the budget to H vectors, where H is the number of transformer heads. The selected activations are used to construct the reduced optimization problem, which we solve using the ADMM procedure in Section 3.3. To assess computational efficiency, we report total rotation-calibration run-time and peak GPU memory in Table 1, with a more detailed breakdown in Table 4. Memory is reported in GiB.

Results. Table 1 shows that ThinQuant improves or closely matches the performance of DartQuant across all reported evaluation metrics on LLaMA-2 and LLaMA-3 with substantially less calibration time. On LLaMA-3-70B, ThinQuant reduces WikiText-2 perplexity from 7.55 to 5.63 and C4 perplexity from 15.69 to 10.34, while reducing total rotation-calibration time from 111 minutes to under 12 minutes. Moreover, Table 4 in the appendix shows that ThinQuant requires several orders of magnitude less disk space than DartQuant. DartQuant requires storing large activation tensors, whereas ThinQuant selects and retains only a tiny subset of activations for rotation calibration. For LLaMA-3-70B, this reduces the required disk space from 960 GiB to 0.51 GiB. Finally, Table 2 shows that ThinQuant scales to LLaMA-3.1-405B, completing rotation calibration in just over 2 hours on a

single H200 GPU. Under W4A4 quantization, replacing QuaRot's fixed Hadamard rotations with ThinQuant reduces WikiText-2 perplexity from 5.82 to 3.10 with GPTQ and from 3.48 to 2.97 with GPTAQ, using the same asymmetric quantization setting as Li et al. (2025). We provide additional experiments on W4A4 including a comparison to OmniQuant in Table 5.

		Quantization performance			Rotation calibration cost	
Model	**Method**	**Wiki-2 ↓**	**C4 ↓**	**0-shot ↑**	**Runtime ↓**	**Peak GPU ↓**
LLaMA-3 8B	Dense FP16	6.14	9.44	65.91		
	ThinQuant	7.91	12.83	**59.70**	**1:56**	**18.5**
	DartQuant	8.22	13.37	59.00	19:03	37.1
	SpinQuant	**7.67**	**12.69**	59.62	53:35	20.7
LLaMA-3 70B	Dense FP16	2.86	7.17	72.70		
	ThinQuant	5.63	10.34	65.83	**11:57**	**31.9**
	DartQuant	7.55	15.69	58.64	111:00	134.9
	SpinQuant	**5.05**	**9.71**	**67.87**	16.7h*	239.0*
LLaMA-2 7B	Dense FP16	5.47	7.26	61.16		
	ThinQuant	6.18	**8.41**	**57.28**	**1:43**	18.2
	DartQuant	6.33	8.63	56.39	20:25	75.1
	SpinQuant	**6.12**	8.49	56.98	49:33	**17.8**
LLaMA-2 13B	Dense FP16	4.88	6.73	64.30		
	ThinQuant	5.47	7.67	61.10	**2:51**	**21.9**
	DartQuant	5.51	7.71	61.35	31:42	51.1
	SpinQuant	**5.35**	**7.61**	**61.73**	113:32	31.5
LLaMA-2 70B	Dense FP16	3.32	5.71	69.52		
	ThinQuant	3.91	6.24	66.92	**11:42**	**33.2**
	DartQuant	3.91	6.22	67.18	99:57	73.1
	SpinQuant	**3.85**	**6.19**	**67.62**	17.2h*	224.3*

Table 1: W4A4KV4 quantization performance and rotation calibration cost. The 0-shot column averages nine downstream tasks. Dense FP16 denotes the unquantized reference model. Calibration-cost columns apply to the quantized methods. Runtime measures total rotation-calibration time for ThinQuant, DartQuant, and SpinQuant. GPU denotes peak GPU process memory during calibration, in GiB. Times are in `mm:ss` unless marked in hours. *The 70B SpinQuant runs used 4 GPUs, so we report total GPU hours and total GPU memory summed across GPUs. Best quantized results are **bold**.

Method	**LLaMA-3 70B**	**LLaMA-3.1 405B**
Dense FP16	2.86	1.44
GPTQ + QuaRot	6.04	5.82
GPTAQ + QuaRot	5.81	3.48
GPTQ + ThinQuant	5.63	3.10
GPTAQ + ThinQuant	**5.46**	**2.97**

Table 2: Scalability to extremely large architectures. WikiText-2 perplexity (↓) on LLaMA-3-70B and LLaMA-3.1-405B. The LLaMA-3-70B column is W4A4KV4 with symmetric weights and asymmetric activations. The LLaMA-3.1-405B column is W4A4. For the 405B comparison, ThinQuant uses the same asymmetric quantization as GPTAQ Li et al. (2025) for both weights and activations. The best quantized result in each column is **bold**. ThinQuant takes 2 hours 7 minutes for full rotation calibration on LLaMA-3.1-405B. We additionally run W4A4KV4 under GPTQ + ThinQuant for LLaMA-3.1-405B, and report the result in Figure 1.

5 Conclusions and Limitations

ThinQuant combines geometric activation selection with an exact subspace reduction to make rotation learning substantially cheaper. ThinQuant achieves competitive low-bit quantization performance compared to state-of-the-art methods with much lower calibration runtime time, GPU memory, and disk storage, making rotation calibration of LLaMA-3.1-405B possible on a single H200 GPU.

Limitations. We remark that we optimize rotations separately prior to using weight quantization algorithms like GPTQ, though the two could in principle be solved jointly. An interesting research direction is to design dedicated methods that can efficiently optimize rotation and weight quantization simultaneously. Moreover, some recent works have shown that learning rotation and diagonal scaling transforms jointly can improve quantization, though this can require expensive calibration (Hu et al., 2025). We leave scalable and efficient rotation and scaling optimization to future work.

AI USE STATEMENT

Generative AI assisted with code development and (minimally) with manuscript revision. The authors are responsible for the final content and verification of the reported results.

REPRODUCIBILITY STATEMENT

The experimental section specifies model families, quantization formats, calibration data, hyperparameters, and evaluation protocols. We additionally provide more details and ablation studies about the algorithms in the Appendix. We will release the code of our activation-selection procedure and ADMM rotation learning algorithm as well as the experimental setup to reproduce our results following the review of the paper.

REFERENCES

Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *Journal of the ACM*, 51(4):606–635, July 2004. doi: 10.1145/1008731.1008736.

Mohammad Sadegh Akhondzadeh, Aleksandar Bojchevski, Evangelos Eleftheriou, and Martino Dazzi. Kurtail : Kurtosis-based llm quantization, 2025. URL https://arxiv.org/abs/2503.01483.

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://arxiv.org/abs/2404.00456.

Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, MA, 1997.

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

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. URL https://web.stanford.edu/~boyd/papers/admm_distr_stats.html.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2307.13304.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Dario Cordero-Erausquin, Matthieu Fradelizi, and Bernard Maurey. The (B) conjecture for the Gaussian measure of dilates of symmetric convex sets and related problems. *Journal of Functional Analysis*, 214(2):410–427, 2004. doi: 10.1016/j.jfa.2003.12.001.

Anil Damle and Yuekai Sun. A geometric approach to archetypal analysis and nonnegative matrix factorization. *Technometrics*, 59(3):361–370, 2017. doi: 10.1080/00401706.2016.1247017.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale, 2022. URL https://arxiv.org/abs/2208.07339.

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://arxiv.org/abs/2210.17323.

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. A framework for few-shot language model evaluation, December 2023. URL https://zenodo.org/records/10256836.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations - 4th Edition*. Johns Hopkins University Press, Philadelphia, PA, 2013. doi: 10.1137/1.9781421407944. URL https://epubs.siam.org/doi/abs/10.1137/1.9781421407944.

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

Xing Hu, Yuan Cheng, Dawei Yang, Zukang Xu, Zhihang Yuan, Jiangyong Yu, Chen Xu, Zhe Jiang, and Sifan Zhou. Ostquant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting, 2025. URL https://arxiv.org/abs/2501.13987.

Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, and Vladimir Gurvich. Generating all vertices of a polyhedron is hard. *Discrete & Computational Geometry*, 39:174–190, 2008. doi: 10.1007/s00454-008-9050-5.

Yuhang Li, Ruokai Yin, Donghyun Lee, Shiting Xiao, and Priyadarshini Panda. Gptaq: Efficient finetuning-free quantization for asymmetric calibration, 2025. URL https://arxiv.org/abs/2504.02692.

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*, 2024. URL https://arxiv.org/abs/2306.00978.

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.

Ryan Lucas, Mehdi Makni, Xiang Meng, Adam Deng, and Rahul Mazumder. Admm-q: An improved hessian-based weight quantizer for post-training quantization of large language models, 2026. URL https://arxiv.org/abs/2605.11222.

Elizabeth S. Meckes. *The Random Matrix Theory of the Classical Compact Groups*. Cambridge Tracts in Mathematics. Cambridge University Press, 2019.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

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.

Yuantian Shao, Yuanteng Chen, Peisong Wang, Jianlin Yu, Jing Lin, Yiwu Yao, Zhihui Wei, and Jian Cheng. DartQuant: Efficient rotational distribution calibration for LLM quantization. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2511.04063.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355. Association for Computational Linguistics, 2018. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446/.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling, 2023. URL https://arxiv.org/abs/2304.09145.

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://arxiv.org/abs/2211.10438.

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*, pages 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Günter M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer, New York, 1995.

A Proof of Theorem 1

Proof. Throughout the proof, the activations and the subset $\mathcal{I}$ are fixed. Write $q = q(\mathcal{I})$ and $H = \max_{i\in[N]} \|x_i\|_2$. Recall that:

$$K = \operatorname{conv}\{\pm x_i : i \in [N]\}, \qquad K_{\mathcal{I}} = \operatorname{conv}\{\pm x_i : i \in \mathcal{I}\}.$$

For $u \in \mathbb{R}^d$, define:

$$f(u) = h_K(u) - h_{K_{\mathcal{I}}}(u) = \max_{i\in[N]} |u^{\mathsf{T}} x_i| - \max_{i\in\mathcal{I}} |u^{\mathsf{T}} x_i|$$

Since $\mathcal{I} \subseteq [N]$, we have $f(u) \geq 0$. The absolute values also imply that $f(-u) = f(u)$ and $f(tu) = tf(u)$ for every $t \geq 0$. We also have $0 \leq f(u) \leq h_K(u) \leq H\|u\|_2$, so $f(g)$ is square-integrable for $g \sim \mathcal{N}(0, I_d)$.

We first show that the expected gap in objectives is bounded by:

$$\mathbb{E}_R\Big[(F(R) - F_{\mathcal{I}}(R))^2\Big] \leq \mathbb{E}_g[f(g)^2] \tag{11}$$

Write the rows of R as $r_1^{\mathsf{T}}, \ldots, r_d^{\mathsf{T}}$. For every rotation,

$$\begin{aligned} F(R) &= \max_{1\leq k\leq d} [h_{K_{\mathcal{I}}}(r_k) + f(r_k)] \\ &\leq \max_{1\leq k\leq d} h_{K_{\mathcal{I}}}(r_k) + \max_{1\leq k\leq d} f(r_k) \\ &= F_{\mathcal{I}}(R) + \max_{1\leq k\leq d} f(r_k). \end{aligned}$$

Since $F_{\mathcal{I}}(R) \leq F(R)$ and $f \geq 0$, this gives:

$$0 \leq (F(R) - F_{\mathcal{I}}(R))^2 \leq \max_{1\leq k\leq d} f(r_k)^2 \leq \sum_{k=1}^{d} f(r_k)^2 \tag{12}$$

Since $d \geq 2$, if $R \sim \operatorname{Haar}(\mathrm{SO}(d))$, each row r_k is marginally uniform on the unit sphere $\mathbb{S}^{d-1}$ (Meckes, 2019). Let v be uniform on $\mathbb{S}^{d-1}$. Taking expectations in (12) therefore gives:

$$\mathbb{E}_R\Big[(F(R) - F_{\mathcal{I}}(R))^2\Big] \leq d\,\mathbb{E}_v[f(v)^2]. \tag{13}$$

Now take $g \sim \mathcal{N}(0, I_d)$ and write $\rho = \|g\|_2$ and $v = g/\|g\|_2$. Then v is uniform on the sphere and independent of ρ (Vershynin, 2018). Moreover,

$$\mathbb{E}[\rho^2] = \mathbb{E}\|g\|_2^2 = \sum_{j=1}^{d} \mathbb{E}[g_j^2] = d$$

Since $f(tu) = tf(u)$ for every $t \geq 0$ and ρ and v are independent:

$$\mathbb{E}_g[f(g)^2] = \mathbb{E}_{\rho,v}[\rho^2 f(v)^2] = \mathbb{E}[\rho^2]\mathbb{E}_v[f(v)^2] = d\,\mathbb{E}_v[f(v)^2]$$

Combining this with (13) proves (11).

It remains to bound the right-hand side of (11). We do this by establishing two variance bounds:

$$(1-q)\mathbb{E}_g[f(g)^2] \leq \operatorname{Var}_g(f(g)) \leq 2H^2 q.$$

For $q < 1$, the lower bound will allow us to bound the second moment by $\operatorname{Var}_g(f(g))/(1-q)$. The upper bound will then give an estimate in terms of H and q.

We first establish the lower bound by relating q to the probability that $f(g)$ is nonzero. For distinct indices i, j, a tie $|g^{\mathsf{T}} x_i| = |g^{\mathsf{T}} x_j|$ can occur only if:

$$g^{\mathsf{T}}(x_i - x_j) = 0 \qquad \text{or} \qquad g^{\mathsf{T}}(x_i + x_j) = 0.$$

Both vectors $x_i - x_j$ and $x_i + x_j$ are nonzero by the identification of activations up to sign. Each equality therefore describes a proper hyperplane, which has probability zero under the standard

Gaussian. There are only finitely many pairs of indices, so the maximizing index is unique almost surely. Whenever this unique maximizing index belongs to $\mathcal{I}$, the two support values coincide and $f(g) = 0$. Whenever it does not belong to $\mathcal{I}$, every selected activation has a strictly smaller absolute projection, and hence $f(g) > 0$. It follows that:

$$\Pr[f(g) > 0] = \Pr\left[i^\star(g) \notin \mathcal{I}\right] = q(\mathcal{I}). \tag{14}$$

The function $f(g)$ is zero outside the event $\{f(g) > 0\}$, which has probability q by (14). Cauchy–Schwarz therefore gives:

$$\begin{aligned}\left(\mathbb{E}_g[f(g)]\right)^2 &= \left(\mathbb{E}_g\left[f(g)\mathbf{1}_{\{f(g)>0\}}\right]\right)^2 \\ &\le \mathbb{E}_g[f(g)^2]\,\mathbb{E}_g\left[\mathbf{1}^2_{\{f(g)>0\}}\right] \\ &= q\,\mathbb{E}_g[f(g)^2]\end{aligned} \tag{15}$$

Using the definition of variance and (15), we obtain the required lower bound:

$$\begin{aligned}\mathrm{Var}_g(f(g)) &= \mathbb{E}_g[f(g)^2] - \left(\mathbb{E}_g[f(g)]\right)^2 \\ &\ge (1-q)\mathbb{E}_g[f(g)^2].\end{aligned} \tag{16}$$

We now establish the upper bound on the variance. We use the strengthened Gaussian Poincaré inequality for functions orthogonal to constants and linear functions (Cordero-Erausquin et al., 2004). Evenness ensures orthogonality to linear functions, but does not imply a zero mean. To satisfy both conditions, we apply the inequality to the function:

$$\widetilde{f}(u) = f(u) - \mathbb{E}_g[f(g)]$$

By construction, $\mathbb{E}_g[\widetilde{f}(g)] = 0$. Moreover, subtracting a constant preserves evenness, so symmetry gives:

$$\mathbb{E}_g[g_j \widetilde{f}(g)] = 0, \qquad j = 1, \ldots, d.$$

Thus, centering removes the constant component, while evenness eliminates the linear component. Subtracting a constant also leaves the gradient unchanged, $\nabla\widetilde{f} = \nabla f$ almost everywhere. The strengthened inequality therefore gives:

$$\begin{aligned}\mathrm{Var}_g(f(g)) &= \mathbb{E}_g[\widetilde{f}(g)^2] \\ &\le \frac{1}{2}\mathbb{E}_g\|\nabla\widetilde{f}(g)\|_2^2 = \frac{1}{2}\mathbb{E}_g\|\nabla f(g)\|_2^2.\end{aligned} \tag{17}$$

To bound the right-hand side of (17), we next control the gradient of f. We first show that f is $2H$-Lipschitz, which implies $\|\nabla f(u)\|_2 \le 2H$ wherever f is differentiable. Since $f = h_K - h_{K_\mathcal{I}}$, it suffices to show that each support function is H-Lipschitz. Indeed, for any $u, v \in \mathbb{R}^d$,

$$\begin{aligned}|h_K(u) - h_K(v)| &\le \max_{i\in[N]} \left||u^\mathsf{T}x_i| - |v^\mathsf{T}x_i|\right| \\ &\le \max_{i\in[N]} |(u-v)^\mathsf{T}x_i| \\ &\le H\|u-v\|_2.\end{aligned}$$

The same argument applies to $h_{K_\mathcal{I}}$. By the triangle inequality, their difference therefore satisfies:

$$|f(u) - f(v)| \le 2H\|u-v\|_2 \tag{18}$$

Both support functions are maxima of finitely many linear functions. Their difference f is therefore piecewise linear and differentiable almost everywhere. At any differentiability point u satisfying $f(u) = 0$, the nonnegativity of f implies that u is a global minimum. Hence $\nabla f(u) = 0$. At every other differentiability point, (18) implies $\|\nabla f(u)\|_2 \le 2H$. Thus,

$$\|\nabla f(u)\|_2^2 \le 4H^2\mathbf{1}_{\{f(u)>0\}}$$

Since the Gaussian measure is absolutely continuous, (14) gives:

$$\mathbb{E}_g\|\nabla f(g)\|_2^2 \le 4H^2 q \tag{19}$$

Applying (17) and then (19), we obtain the required upper bound:

$$\mathrm{Var}_g(f(g)) \le 2H^2 q \tag{20}$$

We now use these two variance bounds to control the second moment required by (11). Suppose first that $q < 1$. Rearranging the lower bound (16) gives the first inequality below, and substituting the upper bound (20) gives the second:

$$\mathbb{E}_g[f(g)^2] \le \frac{\mathrm{Var}_g(f(g))}{1-q} \le \frac{2H^2 q}{1-q} \tag{21}$$

Substituting (21) into (11) therefore yields:

$$\mathbb{E}_R\Big[\big(F(R) - F_{\mathcal{I}}(R)\big)^2\Big] \le \mathbb{E}_g[f(g)^2] \le \frac{2H^2 q}{1-q} \tag{22}$$

Cauchy–Schwarz, followed by (22), then gives:

$$\mathbb{E}_R[F(R) - F_{\mathcal{I}}(R)] \le \Big(\mathbb{E}_R\Big[\big(F(R) - F_{\mathcal{I}}(R)\big)^2\Big]\Big)^{1/2} \le H\sqrt{\frac{2q}{1-q}} \tag{23}$$

For every $R \in \mathrm{SO}(d)$ and every activation,

$$\|Rx_i\|_\infty \le \|Rx_i\|_2 = \|x_i\|_2 \le H$$

Thus:

$$0 \le F(R) - F_{\mathcal{I}}(R) \le H \tag{24}$$

If $0 \le q \le 1/3$, then $1 - q \ge 2/3$, and hence:

$$\frac{2q}{1-q} \le 3q.$$

Equation (23) therefore gives:

$$\mathbb{E}_R[F(R) - F_{\mathcal{I}}(R)] \le H\sqrt{3q}.$$

If $1/3 \le q \le 1$, then $\sqrt{3q} \ge 1$, so (24) gives:

$$\mathbb{E}_R[F(R) - F_{\mathcal{I}}(R)] \le H \le H\sqrt{3q}.$$

Substituting $q = q(\mathcal{I})$ proves (5). □

B Reduced ADMM updates

Let $C = C_d$ and let Λ, Γ be scaled dual variables for $VS - W = 0$ and $CW - Z = 0$. With $V \in \mathrm{St}(d, r)$, we write the augmented Lagrangian of (10) as:

$$\mathcal{L}_\rho = \|Z\|_4^4 + \frac{\rho}{2}\|VS - W + \Lambda\|_F^2 + \frac{\rho}{2}\|CW - Z + \Gamma\|_F^2 - \frac{\rho}{2}\big(\|\Lambda\|_F^2 + \|\Gamma\|_F^2\big) \tag{25}$$

Holding ρ fixed within each sweep, we update

$$Z^+ = \mathrm{prox}_{\|\cdot\|_4^4/\rho}(CW + \Gamma), \tag{26}$$

$$W^+ = (I_d - \tfrac{1}{2}C)\big(VS + \Lambda + C(Z^+ - \Gamma)\big), \tag{27}$$

$$V^+ = PQ^\mathsf{T}, \quad P\Sigma Q^\mathsf{T} = \mathrm{svd}_{\mathrm{thin}}\big((W^+ - \Lambda)S^\mathsf{T}\big), \tag{28}$$

$$\Lambda^+ = \Lambda + V^+S - W^+, \qquad \Gamma^+ = \Gamma + CW^+ - Z^+. \tag{29}$$

For an input entry a, the proximal value is the unique real solution of $4z^3 + \rho(z - a) = 0$. The W-update uses $(I_d + C)^{-1} = I_d - C/2$, while the V-update is an orthogonal Procrustes problem because $\|VS\|_F^2 = \|S\|_F^2$ on $\mathrm{St}(d, r)$.

Penalty selection. For R_1, we initialize $\rho_0 = 10\|CU\|_F^2/(dB)$, ten times the mean squared entry of the centered activation matrix. We then adapt ρ every 20 iterations by primal–dual residual balancing (Boyd et al., 2011). Let:

$$r_{\mathrm{p}} = \max\big(\|W^+ - V^+S\|_F,\ \|CW^+ - Z^+\|_F\big),$$
$$r_{\mathrm{d}} = \rho\max\big(\|W^+ - W\|_F,\ \|Z^+ - Z\|_F\big).$$

At each adjustment step, set:

$$\rho^+ = \begin{cases} 1.25\,\rho, & r_{\mathrm{p}} > 10\,r_{\mathrm{d}}, \\ \rho/1.25, & r_{\mathrm{d}} > 10\,r_{\mathrm{p}}, \\ \rho, & \text{otherwise}, \end{cases}$$

and clip the result to $[10^{-8}, \max(10\rho_0, 10^{-3})]$. The local R_2 problems are generally well behaved in our experiments, so each R_2 solve uses a fixed $\rho = 10$.

Iteration counts and stopping. Each solve alternates one R_1 problem with $R_{2,\ell}$ problem per layer. R_1 runs for at most 1000 iterations and for at least 300. After iteration 300 it stops when both the primal and dual residuals are below 10^{-4}:

$$\max\big(\|W - VS\|_F,\ \|CW - Z\|_F\big) < 10^{-4}, \qquad \rho\max\big(\|W^+ - W\|_F,\ \|Z^+ - Z\|_F\big) < 10^{-4}.$$

Each R_2 solve runs for 300 iterations. In every case the returned factor is the feasible iterate with the smallest objective value.

C Ablations and Additional Experiments

Activation selection. Table 3 compares four selection rules on LLaMA-3-70B under W4A4KV4 quantization, keeping the activation budget and subsequent rotation optimization fixed. Algorithm 1 achieves a WikiText-2 perplexity of 5.63, compared with 6.94 when selecting the largest ℓ_∞ activations after the initial Hadamard rotation and 7.00 when selecting them in the original basis. Uniformly sampling the same number of activations performs substantially worse, with a perplexity of 29.51. These results show that the choice of calibration activations is important at such a small budget.

Global and local rotations. The optimization ablation in Table 3 relates to the contribution of learning the global residual rotation R_1 and the local rotations R_2. Optimizing only R_1, while leaving R_2 at initialization, achieves a perplexity of 5.69, close to the 5.63 obtained when both are optimized. Optimizing only R_2, while keeping R_1 at its initial Hadamard rotation, gives a perplexity of 6.01. Thus, learning the global residual basis is sufficient to approach the performance of the full method in this setting, while also optimizing the local rotations provides a further improvement.

	Selection				**Optimization**		
	Algorithm 1	$\mathrm{Top}_B\ (\|R_0x\|_\infty)$	$\mathrm{Top}_B\ (\|x\|_\infty)$	Random B	R_1, R_2	R_1 only	R_2 only
Wiki-2 ppl ↓	**5.63**	6.94	7.00	29.51	**5.63**	5.69	6.01

Table 3: LLaMA-3 70B W4A4KV4 ablations. **Selection**: all four columns optimize the same ThinQuant rotation and differ only in how the B tokens are chosen. The first column is the Gaussian direction procedure of Algorithm 1. $\mathrm{Top}_B(\|R_0x\|_\infty)$ scores the tokens after the initial Hadamard R_0. $\mathrm{Top}_B(\|x\|_\infty)$ keeps the tokens of largest residual ℓ_∞ norm in the original basis. Random samples the same budget uniformly. **Optimization**: we compare keeping only the global rotation (R_1) or only the local rotatons (R_2). R_1, R_2 is the full Llama-3-70B run. R_1 only optimizes the residual rotation and leaves R_2 at initialization. R_2 only optimizes this rotation and leaves R_1 at the initial Hadamard. Best results in each block are **bold**.

Calibration cost breakdown. Table 4 separates rotation optimization from the total calibration procedure and reports the associated disk storage. On LLaMA-3-70B, ThinQuant reduces optimization time from 74 minutes 46 seconds to 1 minute 22 seconds, approximately a 55× speedup over DartQuant. The total calibration time decreases from 111 minutes to 11:57 minutes, approximately a

$10\times$ speedup. The smaller reduction in total time reflects the cost of activation collection, selection, and data preparation, which together account for most of ThinQuant's calibration time. Across all five models, ThinQuant also reduces disk storage by more than three orders of magnitude. For example, LLaMA-3-70B requires 0.51 GiB rather than 960 GiB. These savings arise from not needing to store the large calibration set used by DartQuant.

Cost	Method	LLaMA-3 8B	LLaMA-3 70B	LLaMA-2 7B	LLaMA-2 13B	LLaMA-2 70B
Optimization	ThinQuant	**23.4s**	**1:22**	**22.0s**	**34.5s**	**1:31**
	DartQuant	12:14	74:46	12:05	21:22	60:47
Total calibration	ThinQuant	**1:56**	**11:57**	**1:43**	**2:51**	**11:42**
	DartQuant	19:03	111:00	20:25	31:42	99:57
Disk storage	ThinQuant	**0.13**	**0.51**	**0.13**	**0.20**	**0.51**
	DartQuant	192	960	193	300	961

Table 4: Breakdown of rotation-calibration cost. Optimization measures rotation optimization only, while total calibration additionally includes activation collection, selection, training-data preparation, and folding the rotation into the weights. Times are in `mm:ss` unless marked in seconds. Disk storage is in GiB. Lower values are better, and best results are **bold**.

		Quantization performance			Calibration cost	
Model	Method	Wiki-2 ↓	C4 ↓	0-shot ↑	Runtime ↓	Peak GPU ↓
LLaMA-3 8B	ThinQuant	**7.75**	**12.55**	**60.87**	**1:56**	18.5
	DartQuant	8.03	13.09	59.98	19:03	37.1
	OmniQuant	71.32	92.86	32.00	54:57	**17.2**
LLaMA-3 70B	ThinQuant	**5.52**	**10.14**	**66.18**	**11:57**	**31.9**
	DartQuant	7.15	14.88	59.05	111:00	134.9
	OmniQuant	482.73	775.47	–	406:49	127.0
LLaMA-2 7B	ThinQuant	**6.12**	**8.32**	**57.58**	**1:43**	18.2
	DartQuant	6.25	8.52	56.93	20:25	75.1
	OmniQuant	11.21	16.05	45.68	53:59	**16.2**
LLaMA-2 13B	ThinQuant	**5.43**	**7.59**	61.25	**2:51**	**21.9**
	DartQuant	5.46	7.63	**61.65**	31:42	51.1
	OmniQuant	10.64	16.62	44.49	93:07	27.3
LLaMA-2 70B	ThinQuant	**3.88**	6.20	67.01	**11:42**	**33.2**
	DartQuant	3.89	**6.19**	**67.31**	99:57	73.1
	OmniQuant	8.43	12.19	49.76	404:08	125.9

Table 5: W4A4 quantization performance and calibration cost. Weights and activations are 4-bit; the KV cache is left at 16 bits. The 0-shot column averages nine downstream tasks. For ThinQuant and DartQuant, runtime is total rotation-calibration time, including activation collection, selection, and training-data preparation. Both reuse the rotations from the W4A4KV4 runs, so those calibration costs match Table 1. OmniQuant runtime is activation-scale collection plus LET/LWC fitting. GPU denotes peak GPU process memory during calibration, in GiB. Times are in `mm:ss`. Best results are **bold**.