Structured Transforms for Low-Overhead Quantization of Language Models

Cherniuk Daria[1] **Alexander Rudikov**[1,2] **Boris Kashin**[3] **Ivan Oseledets**[1,2]

[1]AXXX, [2]Institute of Numerical Mathematics, [3]Steklov Mathematical Institute,

Abstract

We revisit Kashin-decomposition-based weight quantization for large language models and propose an improved algorithm with stronger convergence properties and structured, efficient orthogonal transforms. The method retains the core factorization of each weight into two components – one with bounded infinity norm and the other with bounded infinity norm after an orthogonal transformation – but replaces the dense random orthogonal matrix with a sign-randomized Discrete Cosine Transform (DCT), reducing the per-iteration cost from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$. The proposed greedy algorithm with alternating updates guarantees the four-peak distribution required for stable 2-bit clustering of each factor and admits closed-form initialization of cluster centers, removing the multi-restart k-means bottleneck of prior work. Composed with OPTQ-style sequential error compensation and QuIP-style incoherence preprocessing, the resulting JAX pipeline is competitive with OPTQ, QuIP, QuIP-RG, a fine-tuning- and vector-quantization-free variant of QuIP#, AWQ, and OmniQuant at 4-bit per channel on OPT, Llama-2, Llama-3, and Pythia, with favorable wall-clock scaling. The bounded-ℓ_∞ factorization is also notably robust: on stress configurations where QuIP variants diverge to four-digit perplexity (Pythia-6.9B) or abort with NaNs in LDL back-substitution (Mistral-7B), Kashin-DCT remains numerically stable and stays close to FP16 baseline. At inference time, each weight decomposes into two 2-bit factor codes per channel that are structurally suited to native-2-bit hardware.

1 Introduction

Modern Large Language Models (LLMs) achieve their capabilities at the cost of weight tensors that dominate memory bandwidth and footprint at inference time. Post-training quantization (PTQ) compresses these weights to low bit-widths without re-training, making it the practical choice when the training pipeline and large-scale compute are unavailable. Lower inference precision also translates into lower energy draw per query and a correspondingly smaller carbon footprint. In deployment, the dominant scheme remains plain uniform scalar quantization with per-channel or per-group scales [Dettmers et al., 2022], sometimes with a non-uniform grid such as NF4 [Dettmers et al., 2023]. The research literature improves on this baseline first via second-order error compensation [Frantar et al., 2023, Frantar and Alistarh, 2022] and incoherence preprocessing [Chee et al., 2023, Tseng et al., 2024a], and most recently via *vector quantization* (VQ): AQLM [Egiazarian et al., 2024], GPTVQ [van Baalen et al., 2024], and QTIP [Tseng et al., 2024b] currently lead sub-4-bit benchmarks. VQ comes with three practical costs, however: each layer must store learned codebook entries alongside the integer indices, the strongest results require gradient-based fine-tuning, and codebook lookups do not fuse into standard matmul kernels as cleanly as scalar dequantization.

A complementary line of work [Merkulov et al., 2024] approaches low-bit quantization through *Kashin decomposition*: each vectorized weight matrix w is factorized as $w = u + P^T \hat{v}$, where P is

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

a random orthogonal matrix, u and $\hat{v} = Pv$ separately have small infinity norms. Empirically, the components produced by Kashin's greedy algorithm form distributions with four sharp symmetric peaks [Merkulov et al., 2024], which is a near-ideal target for 2-bit clustering. Despite this attractive structure, the prior application of Kashin decomposition to LLMs [Merkulov et al., 2024] suffered from three weaknesses. First, to amortize cost the authors reformulated the algorithm matrix-wise, which dropped the per-vector convergence guarantees of the original theorem [Kashin, 1977]; on a non-trivial fraction of layers the iteration failed to converge. Second, even on layers where the iteration did converge, the four-peak structure that 2-bit clustering relies on was not consistently produced: the joint u-$\hat{v}$ distribution could collapse rather than separate into the four symmetric modes (Figure 3(a)); Third, the four cluster centers were recovered with multi-restart k-means, an unstructured search that dominated quantization wall-clock time.

Contributions. We revisit Kashin-decomposition-based quantization end-to-end and address all three weaknesses, while keeping the analysis at the vector level where rigorous convergence proofs are available.

- **A partitioned greedy algorithm with guaranteed peaks and a structured DCT transform (Section 4).** Building on the recent acceleration of Kashin's theorem in [Kashin et al., 2025], we propose a Greedy Algorithm with Alternating Updates (Algorithm 1) that fixes the order of u- and $\hat{v}$-updates in blocks of four. The schedule guarantees four-peak distributions in both factors, and we prove the conversion guarantees. We further replace the dense random orthogonal Q by a sign-randomized Discrete Cosine Transform with $\mathcal{O}(N \log N)$ cost and zero stored matrix.
- **Closed-form cluster centers (Section 3.1).** The greedy updates change the factors by some value $\pm c_k(r_k)$ at each step, so after two updates the four peak locations are exactly $\pm c_1 \pm c_2$, known analytically from the r_k residual norm. We initialize k-means with these centers and run only a handful of refinement iterations, eliminating multi-restart search and reducing quantization time.
- **Adaptive integration and a JAX pipeline (Section 5).** We compose Kashin decomposition with OPTQ-style sequential error compensation and QuIP-style incoherence preprocessing (Hadamard or Kronecker), and implement the full pipeline in compile-friendly JAX with multi-GPU `pmap` support. On OPT, Pythia, and Llama-family models at 4-bit per channel, the method is competitive with OPTQ, QuIP, QuIP-RG, (fine-tuning-free, vector-quantization-free) QuIP#, AWQ, and OmniQuant on WikiText-2 and C4 perplexity and with the methods in Table 1 on HellaSwag, PiQA, and Winogrande accuracy.

2 Related Work

Quantization techniques for deep neural networks can be broadly categorized into post-training quantization (PTQ) and quantization-aware training (QAT). QAT typically delivers the highest accuracy because it embeds low-precision constraints directly into the optimization loop; however, it also demands extensive GPU time and memory, especially when working with the very large models that stand to benefit most from reduced-precision inference. PTQ, in contrast, converts a pre-trained model to lower bit-widths without additional gradient-based fine-tuning, making it attractive for practitioners who lack the computational resources or training datasets.

The fastest and easiest-to-implement PTQ schemes rely on simple heuristics, such as uniform round-to-nearest or stochastic rounding applied per layer or per row Dettmers et al. [2022], or assume a particular activation/weight distribution, as in NormalFloat 4-bit (NF4) quantization Dettmers et al. [2023]. Activation- and outlier-aware methods reduce the dynamic range that scalar quantization must cover, either by smoothing the activation/weight magnitudes [Xiao et al., 2023], by scaling salient channels [Lin et al., 2024a], by jointly learning weight clipping ranges and equivalent transformations [Shao et al., 2024], or by routing outliers to a dense non-uniform codebook [Kim et al., 2024].

Recent state-of-the-art methods use second-order information to minimize the quantization-induced error in the layer outputs. OPTQ Frantar et al. [2023] sequentially quantizes chunks of layer weights while compensating for the accumulated error through corrections to the yet-to-be-quantized parameters of the layer. QuIP [Chee et al., 2023] shows that OPTQ is a special case of LDLQ adaptive

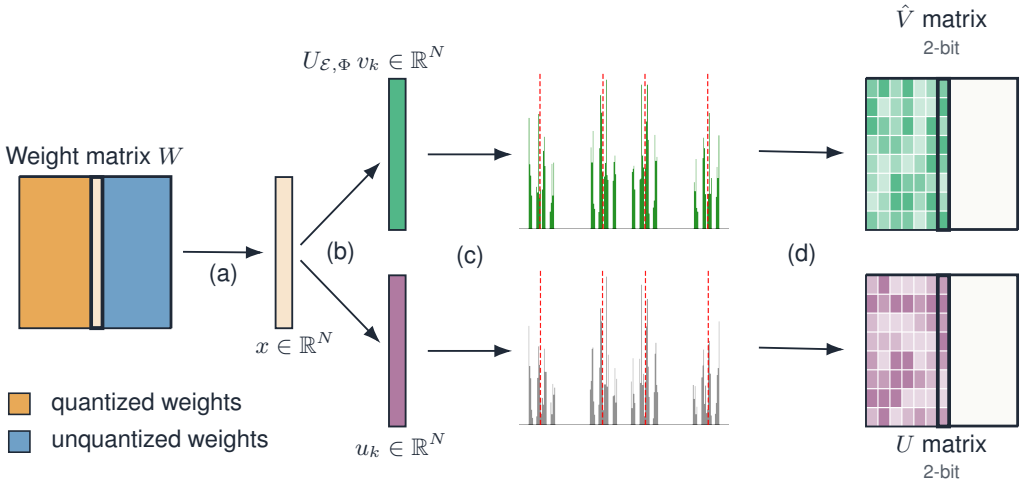

Figure 1: The proposed quantization pipeline consists of several stages: (a) move to the next column to quantize; (b) decompose with the proposed Greedy Algorithm with Alternating Updates (Algorithm 1) into u and $\hat{v}$; (c) quantize each factor to 2 bits via 4-peak clustering; (d) stack the quantized vectors into the output matrices U and $\hat{V}$; then add error compensation to the unquantized weights and return to (a).

quantization, essentially the method introduced in QuIP without incoherence processing, in which the correction to the yet-to-be-quantized rows of the weight matrix is a linear combination of the quantization errors of the already-quantized rows. Authors further propose incoherence pre- and post-processing to bring weight matrices into the conditions of their theorem, which establishes that LDLQ quantization is a lower bound on nearest and stochastic rounding.

QuIP# [Tseng et al., 2024a] replaces multiplication by random orthogonal matrices with randomized Hadamard transforms for incoherence processing, which yields better incoherence properties and faster runtime. However, QuIP# uses vector quantization and resorts to fine-tuning. The same randomized-rotation idea is the basis for a parallel line of activation-quantization methods: QuaRot [Ashkboos et al., 2024] fuses Hadamard rotations into the residual stream so that both weights and activations are quantized in the rotated basis, SpinQuant [Liu et al., 2025] learns the rotation matrices on the calibration distribution, and DuQuant [Lin et al., 2024b] composes block-rotations with permutations to suppress outliers further.

Beyond QuIP#, a substantial line of work pushes weight compression further by switching from scalar to *vector* codebooks. AQLM [Egiazarian et al., 2024] learns an additive composition of small codebooks per group of weights; GPTVQ [van Baalen et al., 2024] extends OPTQ-style error compensation to multi-dimensional codebooks; and QTIP [Tseng et al., 2024b] replaces QuIP#'s lattice codebook with trellis-coded quantization on top of the same incoherence processing. These methods achieve very strong sub-4-bit compression ratios, but share two practical costs: each layer must store learned codebook entries alongside the integer indices, and reaching the reported quality typically requires gradient-based fine-tuning or extensive calibration sweeps. The Kashin decomposition we revisit in this work, by contrast, retains *scalar* 2-bit clusters and incurs only a handful of centroids per column (Section 3.1), with no fine-tuning step.

3 Problem Setting

The work of Kashin [1977] has introduced the following theorem.

Theorem 1. *For each $N = 2, 3, \ldots$ and for each orthogonal transformation $P \in \mathbb{O}^N$, with the exception of a set $V \subset \mathbb{O}^N$, $\quad \mu_H(V) \leqslant 2^{-N}$, the following inequality holds:*

$$\max\left\{\|x\|_1,\, \|Px\|_1\right\} \geqslant c_1 \cdot \sqrt{N}\|x\|_2 \qquad \forall x \in \mathbb{R}^N, \tag{1}$$

where $c_1 > 0$ is an absolute constant.

Using (1), for each $P \notin V$, a greedy algorithm [Temlyakov, 2011] was constructed such that, for every $x \in B_2^N = \{x \in \mathbb{R}^N : \|x\|_2 \leqslant 1\}$, after k steps, it produces vectors u_k and v_k that satisfy

$$\|u\|_\infty \leqslant \frac{c_2}{\sqrt{N}}, \ \|Pv\|_\infty \leqslant \frac{c_2}{\sqrt{N}}, \tag{2}$$
$$\left\|x - u_k - v_k\right\|_2 \leqslant \gamma^k,$$

where $\gamma < 1$ and c_2 are absolute constants. See Appendix B for full algorithm. The algorithm requires storing N^2 entries of the matrix Q and entails a computational complexity of $\mathcal{O}(N^2)$ operations per iteration.

The upper bounds on the infinity norm in (2) naturally lead to using this factorization for quantization of neural network weights, since the most common uniform quantization suffers severely from outliers [Nagel et al., 2021]. An even more interesting phenomenon is that the distributions of both u_k and Pv_k often form four distinctive symmetric peaks [Merkulov et al., 2024], which enables straightforward clustering-based quantization. The former work used this factorization to quantize the weights of an LLM. Their approach reformulated Kashin's theorem for the matrix case but failed to provide a rigorous convergence analysis for that reformulation. Consequently, the approach encountered convergence failures and ill-defined peaks on a fraction of layers. Moreover, the clustering process relied on the k-means algorithm with multiple restarts, which significantly slowed quantization.

In this work, we retain the vector formulation, for which rigorous proofs are available, while alleviating the large size of the matrix Q noted in Merkulov et al. [2024] by using a sign-randomized Discrete Cosine Transform (DCT) as the orthogonal transformation, leveraging the recent acceleration of Kashin et al. [2025]; we then partition the greedy schedule so that the dominant updates land at analytically known centroid locations, which removes the multi-restart k-means step. The proposed Greedy Algorithm with Alternating Updates and its convergence analysis are presented in Section 4; the closed-form k-means initialization derived from this schedule is described in Section 3.1 below. Appendix C ablates each of these three components. Relative to a dense random orthogonal transform, the sign-randomized DCT is $6.63\times$ faster per column and reduces the reconstruction error from 1.73×10^{-4} to 2.57×10^{-7} at the same iteration budget. The block-greedy schedule converges at nearly the same rate as the unpartitioned algorithm while producing the four sharp peaks required in both factors. Finally, the closed-form initialization reduces the relative quantization error from 0.1372 to 0.1209 with a single k-means run, compared with ten random restarts.

We further combine Kashin quantization with adaptive quantization methods that quantize model weights chunk by chunk, compensating for the introduced error by adaptively updating the yet-to-be-quantized weights, in the spirit of OPTQ and QuIP; experimental details are provided in Section 5.

3.1 Closed-form k-means initialization

Previous work [Merkulov et al., 2024] applied k-means clustering to the four-peak distributions of u and Pv with random centroid initialization and multiple restarts ($n_{\text{init}} = 50$ in their reported configuration), which dominated quantization wall-clock time. We exploit the structure of the updates in Algorithm 2: at iteration k, the update of the vector u is $\frac{1}{N}\|r_k\|_1 \operatorname{sign}(r_k)$, which corresponds to adding $\pm c_k(r_k)$ where r_k is a residual at iteration k. After two such updates, the marginal distribution of u contains four peaks at exactly $\pm c_1 \pm c_2$, known analytically from the residual norms. The appearance and the values $\pm c_1 \pm c_2$ of these peaks are a direct consequence of the definition of Algorithm 1 and the strong concentration of the function $\dfrac{\|x\|_{\ell_1^N}}{\sqrt{N}}$ around its median, which converges to $\sqrt{2/\pi}$ as N increases. Because the residual norm decreases rapidly across iterations, these early updates already provide accurate approximations of the peak locations in the final distribution; in practice the first four iterations are sufficient. The fixed schedule of Algorithm 1 (Section 4) ensures that exactly two updates per factor land at the analytic centroids. After decomposition converges, the analytic $\pm c_1 \pm c_2$ values are used to initialize the cluster centers and a small number of k-means refinement iterations are run; no multi-restart outer loop is required. The quantization procedure for $\hat{v}$ follows the same strategy. The closed-form initialization removes the multi-restart loop entirely, reducing the clustering wall-clock time by roughly $10\times$ relative to the multi-restart 2-D k-means baseline of [Merkulov et al., 2024] on medium-sized models.

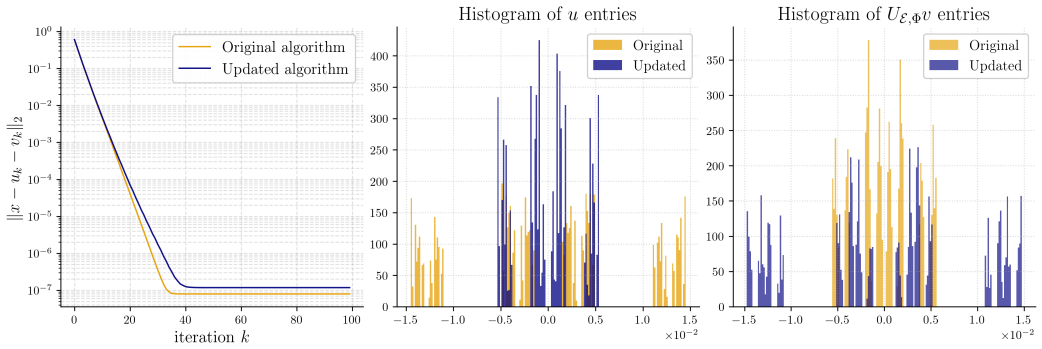

Figure 2: Convergence and factor-distribution comparison on a single random vector $x \in \mathbb{R}^N$ with $N = 10^4$ drawn from $\mathcal{N}(0, I_N)$. Left: residual norm $\|x - u_k - v_k\|_2$ versus the number of greedy updates for the original greedy algorithm of Kashin [1977] (Algorithm 2 in Appendix B, violet) and our partitioned variant of Algorithm 1 (yellow). Center, right: empirical densities of u_k and $U_{\mathcal{E},\Phi}\, v_k$ at convergence; the partitioned schedule produces four sharp symmetric peaks, while the original schedule leaves the second factor unimodal.

4 The Greedy Algorithm with Alternating Updates

4.1 Background

Let $\Phi = \{\phi_j\}_{j=1}^N$ be an orthonormal basis in $\mathbb{R}^N$ such that $\|\phi_j\|_\infty \leqslant K/\sqrt{N}$, $1 \leqslant j \leqslant N$. For the orthonormal DCT-II used throughout this paper[1], $\phi_j(n) = \sqrt{2/N}\cos\big(\pi(2n+1)j/(2N)\big)$ for $j = 0, \dots, N-1$, so $\|\phi_j\|_\infty \leqslant \sqrt{2/N}$ and the constant K in our analysis equals $\sqrt{2}$, i.e. a small absolute constant independent of N. As in Kashin et al. [2025], we consider the orthogonal operator $U_{\mathcal{E},\Phi} = \mathcal{F}_\Phi^{-1}\, T_\mathcal{E}\, \mathcal{F}_\Phi$, where the orthogonal operator $\mathcal{F}_\Phi$ acts in $\mathbb{R}^N$ according to the rule $\mathcal{F}_\Phi\big(\sum_{j=1}^N a_j\phi_j\big) = \{a_j\}_{j=1}^N = a \in \mathbb{R}^N$, let $\mathcal{F}_\Phi^{-1}$ be the inverse of $\mathcal{F}_\Phi$, and let the operator $T_\mathcal{E}$ be defined for a given random set of signs $\mathcal{E} = \{\varepsilon_j\}_{j=1}^N$, $\varepsilon_j = \pm 1$, $1 \leqslant j \leqslant N$ by the relation $T_\mathcal{E}(a) = \{\varepsilon_j a_j\}_{j=1}^N$.

In Kashin et al. [2025], the authors leveraged the fact that for most sign sets $\mathcal{E}$, the following relation holds:

$$\max\Big(\|x\|_1,\, \|U_{\mathcal{E},\Phi}x\|_1\Big) \geqslant \frac{R(N)}{K\sqrt{2}} \qquad \forall x \in \mathbb{R}^N,\ \|x\|_2 = 1, \tag{3}$$

where $R(N) = \dfrac{\sqrt{N}}{c_3\big(\log N\big)^{1/2}\big(\log\log N\big)^3}$, and c_3 is an absolute constant. Throughout the rest of the paper we instantiate $\mathcal{F}_\Phi$ as the orthonormal DCT-II and write $P = U_{\mathcal{E},\Phi}$; with $\mathcal{F}_\Phi^{-1} = \mathcal{F}_\Phi^\top$ the operator P admits the FFT-fast realization $Pz = \mathrm{IDCT}\big(\varepsilon \odot \mathrm{DCT}(z)\big)$ used in Algorithm 1.

The authors of Kashin et al. [2025], using the result from Guédon et al. [2008], propose a greedy algorithm in N-dimensional Euclidean space with the dictionary $S = Q_N \bigcup U_{\mathcal{E},\Phi}Q_N$, where $Q_N \subset \mathbb{R}^N$ is the set of vertices of the cube B_∞^N (i.e., the set of vectors of the form $(\delta_1, \dots, \delta_N)$ with $\delta_i = \pm 1$, $1 \leqslant i \leqslant N$). It is shown that if (3) holds for a sign set $\mathcal{E}$, then for an arbitrary vector $x \in \mathbb{R}^N$ with $\|x\|_2 \leqslant 1$, this greedy algorithm constructs vectors u_k and v_k from $\mathbb{R}^N$ in k steps such that:

$$\begin{aligned} &\|x - u_k - v_k\|_2 \leqslant \big(1 - \alpha(N)\big)^{k/2}, \qquad \alpha(N) = \frac{R^2(N)}{2K^2N} \\ &\max\Big(\|u_k\|_\infty,\, \|U_{\mathcal{E},\Phi}v_k\|_\infty\Big) \leqslant \frac{4c_3^2K^2\log N\big(\log\log N\big)^6}{\sqrt{N}}. \end{aligned} \tag{4}$$

In this paper, based on numerical experiments, we propose a modification of the algorithm from Kashin et al. [2025] that offers several advantages over the original method. Notably, while the

[1]Throughout this paper we use the orthonormal DCT-II (the `norm='ortho'` convention), under which $P_{\mathrm{IDCT}} = P_{\mathrm{DCT}}^\top$

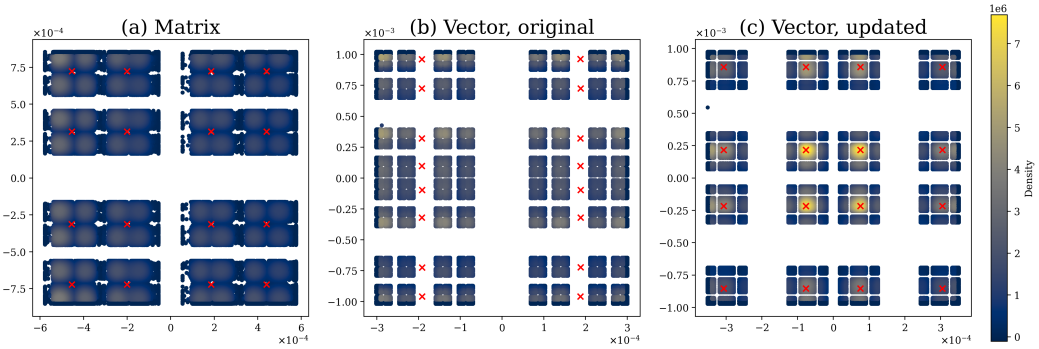

Figure 3: Joint value distribution of u and $\hat{v}$ for the fc2 layer weights of the third decoder block of OPT-125m, decomposed by three algorithms. Red crosses indicate the cluster centers found by k-means after factorization. *(a) Matrix reformulation of Merkulov et al. [2024]*: the matrix-wise greedy iteration drops the per-vector convergence guarantee and the u-$\hat{v}$ joint distribution collapses, leaving cluster centers ill-defined. *(b) Original vector greedy algorithm [Kashin, 1977] (Algorithm 2)*: per-vector convergence is restored but the $\hat{v}$-axis remains spread-out, so 2-bit clustering of $\hat{v}$ is degenerate. *(c) Proposed Algorithm 1*: the partitioned schedule and DCT-based P produce four-peaked marginals along both axes, yielding well-separated cluster centers.

modification provides practical benefits, the resulting theoretical bound on the convergence rate for the new algorithm is slightly weaker than the estimate provided in Kashin et al. [2025].

4.2 Proposed Algorithm

The proposed modification retains the dictionary $S = Q_N \cup U_{\mathcal{E},\Phi} Q_N$ of Kashin et al. [2025], but replaces the single-step global greedy selection with a *partitioned* scheme that alternates between the two halves in blocks of four greedy steps: within each block, the first two atomic greedy steps draw from Q_N (and accumulate into u), while the last two draw from $U_{\mathcal{E},\Phi} Q_N$ (and accumulate into $\hat{v} = U_{\mathcal{E},\Phi}\, v$); see Algorithm 1. The slowdown $\beta(N) = \alpha^2(N)/36$ established in Proposition 1 is the explicit price of fixing this schedule rather than letting each step adaptively choose the better half as in Kashin et al. [2025]; in exchange, the dominant updates of u and $\hat{v}$ land at the analytic centroid locations $\pm c_1 \pm c_2$ used by the closed-form k-means initialization of Section 3.1. By symmetry of the convergence argument the reverse schedule $(U_{\mathcal{E},\Phi} Q_N, U_{\mathcal{E},\Phi} Q_N, Q_N, Q_N)$ yields the same residual norm at every block of 4 iterations.

In the computational experiments, the sign pattern ε was kept fixed throughout Algorithm 1, but it seems promising to consider the case in which the sign pattern is resampled independently at each iteration.

4.3 Convergence Analysis

We now estimate the convergence rate of Algorithm 1. Proposition 1 below shows that one block of four atomic greedy steps contracts the residual norm by a factor $(1 - \beta(N))^{1/2}$ with $\beta(N) = \alpha^2(N)/36$; iterating yields the geometric residual decay of Theorem 2. Reaching a target residual norm therefore takes polylogarithmic factor of iteration steps of the unconstrained algorithm of Kashin et al. [2025].

Proposition 1. *Let $\alpha(N) = R^2(N)/2K^2N$ and $\beta(N) = \alpha^2(N)/36$. For any $x \in \mathbb{R}^N$ with $\|x\|_2 \leqslant A$, one block of the partitioned greedy algorithm produces a residual r_1 satisfying*

$$\|r_1\|_2 \leqslant A\big(1 - \beta(N)\big)^{1/2}. \tag{5}$$

Algorithm 1 Greedy Algorithm with Alternating Updates (DCT-based realization).

Input: Vector $x \in \mathbb{R}^N$, sign mask $\varepsilon \in \{-1, +1\}^N$, number of blocks k
Output: Vectors $u_k, \hat{v}_k \in \mathbb{R}^N$ such that

$$x \;\approx\; u_k + P\hat{v}_k \;=\; u_k + v_k,$$

where the orthogonal operator $P : \mathbb{R}^N \to \mathbb{R}^N$ is defined by

$$Pz \;:=\; \mathrm{IDCT}\big(\varepsilon \odot \mathrm{DCT}(z)\big).$$

Initialize $u_0 \leftarrow \mathbf{0}_N,\ \hat{v}_0 \leftarrow \mathbf{0}_N,\ r_0 \leftarrow x$
for $t = 0, 1, \ldots, k-1$ **do**
  $\rho \leftarrow r_t, \quad u \leftarrow u_t, \quad \hat{v} \leftarrow \hat{v}_t$
  for $j = 1, 2$ **do**
    // two greedy steps from Q_N
    $\Delta u \leftarrow \mathrm{sign}(\rho) \cdot \dfrac{\|\rho\|_1}{N}$
    $u \leftarrow u + \Delta u, \qquad \rho \leftarrow \rho - \Delta u$
  end for
  for $j = 3, 4$ **do**
    // two greedy steps from $U_{\mathcal{E},\Phi} Q_N$
    $\Delta\hat{v} \leftarrow \left(\mathrm{sign}(P\rho) \cdot \dfrac{\|P\rho\|_1}{N}\right)$
    $\hat{v} \leftarrow \hat{v} + \Delta\hat{v}, \qquad \rho \leftarrow \rho - P\,\Delta\hat{v}$
  end for
  $u_{t+1} \leftarrow u, \quad \hat{v}_{t+1} \leftarrow \hat{v}, \quad r_{t+1} \leftarrow \rho$
end for
Return: $u_k,\ \hat{v}_k,\ r_k$ (and $v_k = P\hat{v}_k$ if needed, by symmetry $P^2 = I$)

Theorem 2. *Let $\alpha(N) = R^2(N)/2K^2N$ and $\beta(N) = \alpha^2(N)/36$. For any $x \in \mathbb{R}^N$ with $\|x\|_2 \leqslant 1$, the partitioned greedy algorithm after k blocks of iterations constructs $u_k, v_k \in \mathbb{R}^N$ such that*

$$\begin{aligned} \big\|x - u_k - v_k\big\|_2 &\leqslant \big(1 - \beta(N)\big)^{k/2}, \\ \max\Big(\big\|u_k\big\|_\infty,\, \big\|U_{\mathcal{E},\Phi}\, v_k\big\|_\infty\Big) &\leqslant \frac{4}{\beta(N)\sqrt{N}} \;=\; \frac{c_4\, K^4\, (\log N)^2\, (\log\log N)^{12}}{\sqrt{N}}, \end{aligned} \tag{6}$$

where c_4 is an absolute constant.

Proof sketch. The inequality (3) guarantees that, for any residual ρ, at least one of the two dictionary halves Q_N or $U_{\mathcal{E},\Phi}Q_N$ is the "good" half, in the sense that a single greedy step from it contracts the squared ℓ_2 norm by a factor $(1 - \alpha(N))$. The unconstrained algorithm of Kashin et al. [2025] picks the good half adaptively at every step. The partitioned algorithm cannot: it is forced to spend steps $1, 2$ on Q_N regardless. Proposition 1 reduces to two cases. **(i)** If Q_N *is* the good half for the block's input residual ρ_0, step 1 alone already gives $\|\rho_1\|_2 \leqslant A\sqrt{1-\alpha}$, which is stronger than the claim; steps 2–4 are non-expansive. **(ii)** If $U_{\mathcal{E},\Phi}Q_N$ is the good half, steps $1, 2$ on Q_N are "wasted" but, by orthogonality of consecutive greedy updates, their combined contribution has norm $\leqslant A\sqrt{2\beta(N)}$. Step 3 from $U_{\mathcal{E},\Phi}Q_N$ is then at least as good as the best one-shot approximation $\lambda^* w^*$ of ρ_0 from that half, which by the one-step estimate satisfies $\|\rho_0 - \lambda^* w^*\|_2 \leqslant A\sqrt{1-\alpha}$. The triangle inequality combines these into $\|\rho_3\|_2 \leqslant A\sqrt{1-\alpha} + A\sqrt{2\beta} \leqslant A\sqrt{1-\beta}$, where the final step is exactly the algebraic identity that fixes the constant $\beta(N) = \alpha^2(N)/36$. Theorem 2 then iterates Proposition 1 over k blocks for the ℓ_2 bound, and bounds $\|u_k\|_\infty$ and $\|U_{\mathcal{E},\Phi}v_k\|_\infty$ by summing the per-step coefficients $|\lambda_j^{(\ell)}| \leqslant \|r_{j-1}\|_2/\sqrt{N}$ across blocks; the geometric residual decay makes that sum a convergent series of order $1/(\beta(N)\sqrt{N})$. Full proofs of Proposition 1 and Theorem 2 can be found in Appendix F.

5 Experiments

We conduct experiments on the OPT [Zhang et al., 2022], Llama-2 [Touvron et al., 2023], Llama-3 [Grattafiori et al., 2024], Mistral [Jiang et al., 2023], and Pythia [Biderman et al., 2023] families of models. We use WikiText-2 [Merity et al., 2017] and C4 [Raffel et al., 2020] for measuring perplexity, and HellaSwag [Zellers et al., 2019], PiQA [Bisk et al., 2020], and Winogrande [Sakaguchi et al., 2021] for zero-shot accuracy. All evaluations are run through the `lm-evaluation-harness` framework[2] for reproducible and reliable results, with maximum sequence length set to 2048. For the Hessian-based methods in Table 1, the per-layer Hessian $H = X^\top X$ is estimated on 1000 sequences of length 2048 drawn from the WikiText-2 (`wikitext-2-raw-v1`) training split, and the same calibration set is used across these baselines. Unless stated otherwise, each row in Table 1 reports mean $\pm$ standard deviation over three random seeds; the seed determines the calibration data shuffle for all methods, the sign set ε for our Kashin-DCT pipeline, and the random incoherence rotation for any method that uses one. All quantization and evaluation runs are performed on a single NVIDIA H100 GPU; per-method end-to-end quantization wall-clock times are reported in Appendix E, and an inference-time analysis for the Kashin representation is given in Appendix D.

First, we demonstrate the advantages of our Greedy Algorithm with Alternating Updates over the original greedy algorithm. Figure 2 shows that, on a random vector drawn from $\mathcal{N}(0, 1)$, Algorithm 1 attains nearly the same convergence rate as the original vector greedy algorithm while producing markedly sharper peak definition.

Second, we show that our method does not fail on layers where the previous Kashin-decomposition-based approach encountered problems. Figure 3 compares the joint u and $\hat{v}$ distributions for the matrix reformulation of Merkulov et al. [2024], the original greedy algorithm, and our updated algorithm. The proposed approach is the only one that produces distinctive peaks in both the u and Pv value distributions, enabling stable cluster quantization.

Finally, we compare our quantization pipelines with other PTQ methods, namely Round-To-Nearest (RTN), OPTQ [Frantar et al., 2023], QuIP [Chee et al., 2023] (with two decomposition variants, LDLQ and LDLQ-RG), and a fine-tuning- and vector-quantization-free variant of QuIP# [Tseng et al., 2024a]. For all methods, the per-output-channel target is 4 bits per weight. The rows of Table 1 correspond to the following configurations: **QuIP** = LDLQ with Kronecker-of-rotations incoherence preprocessing; **QuIP-RG** = LDLQ-RG with Kronecker-of-rotations incoherence preprocessing; **QuIP#** = LDLQ-RG with a randomized Hadamard transform for incoherence preprocessing, with end-to-end fine-tuning and the E8 lattice vector codebook of the original method *disabled*, since we only consider PTQ value quantization methods that do not require fine-tuning. **Kashin-DCT+K** and **Kashin-DCT+H** are Kashin decomposition with OPTQ-style sequential error compensation and Kronecker and randomized-Hadamard incoherence preprocessing, respectively. Results are presented in Table 1.

To broaden the comparison to activation-aware PTQ, Table 2 reports perplexity for AWQ [Lin et al., 2024a] and OmniQuant [Shao et al., 2024] alongside our two strongest configurations. All methods use per-output-channel 4-bit weights and are evaluated through the same `lm-evaluation-harness` setup. AWQ uses its standard activation-aware search with full-column groups (`q_group_size` set to -1), 512 Pile calibration samples of length 512, and a different calibration shuffle for each seed. OmniQuant uses W4A16 learnable weight clipping with equivalent transformations disabled, calibrated on 128 WikiText-2 samples for 20 epochs. Each entry is the mean $\pm$ sample standard deviation over three calibration seeds. Kashin-DCT obtains the lowest perplexity on eight of the ten model–dataset pairs; OmniQuant performs best on both Pythia-1.4B datasets.

We note that the LDLQ + Kronecker configurations (rows "QuIP" / "QuIP-RG") exhibit larger variance and a perplexity regression on Pythia-1.4B and Llama-2-7B (marked † in Table 1), and the QuIP# row, which uses the randomized Hadamard variant, recovers the expected behavior. This QuIP instability also appears on Pythia-6.9B, where all three QuIP variants we ran fail catastrophically – QuIP and QuIP-RG diverge to > 2000 WikiText-2 PPL, and even QuIP# blows up to ~ 325 PPL – while OPTQ stays close to its baseline at 12.02 PPL. "Kashin-DCT+H" recovers to 20.6 ± 1.6 WikiText-2 PPL, more than an order of magnitude better than QuIP# at the same bit budget; full per-task numbers are reported in Appendix A (Table 3). On Mistral-7B v0.1 the failure is sharper still: all four QuIP variants abort with NaN in the LDL back-substitution on the SwiGLU `mlp.down_proj`

[2] `https://github.com/EleutherAI/lm-evaluation-harness`

Table 1: Perplexity (PPL) on WikiText-2 / C4 and accuracy (acc_norm for HellaSwag and PiQA, acc for Winogrande) at 4-bit per-output-channel weight quantization. Best quantization result per column in **bold**, second best underlined (FP16 and RTN excluded from ranking). Context length 2048; each cell is mean ± std over 3 random seeds (except for FP16 and RTN). †LDLQ (QuIP) and LDLQ-RG (QuIP-RG) are unstable on Pythia-1.4B and Llama-2-7B in our reruns; see Section 5. Pythia-6.9B stress test in Appendix A.

Method	Pythia-1.4B					OPT-1.3B				
	Wiki-2 ↓	C4 ↓	Hella ↑	PiQA ↑	Wino ↑	Wiki-2 ↓	C4 ↓	Hella ↑	PiQA ↑	Wino ↑
FP16	14.72	38.76	52.12	71.16	57.30	16.47	39.41	41.36	71.00	59.75
RTN	18.73	49.99	49.72	69.42	55.33	29.46	73.25	34.31	67.52	53.59
OPTQ	**16.12 ± 0.01**	43.47 ± 0.02	**50.94 ± 0.27**	70.26 ± 0.77	56.33 ± 0.63	17.63 ± 0.01	42.45 ± 0.17	40.46 ± 0.14	70.51 ± 0.28	58.51 ± 0.40
QuIP	19.76† ± 0.44	52.36† ± 0.94	49.84† ± 0.03	70.06† ± 0.98	56.75† ± 0.49	17.91 ± 0.08	42.83 ± 0.17	40.43 ± 0.17	**70.82 ± 0.41**	58.69 ± 0.72
QuIP-RG	20.84† ± 0.39	54.71† ± 0.79	49.89† ± 0.43	69.71† ± 0.36	56.64† ± 0.91	17.89 ± 0.06	42.75 ± 0.16	40.19 ± 0.06	70.51 ± 0.85	**59.27 ± 0.99**
QuIP #	17.25 ± 0.18	45.92 ± 0.49	50.35 ± 0.30	**70.57 ± 0.30**	**57.43 ± 0.16**	17.83 ± 0.11	42.75 ± 0.36	40.53 ± 0.12	70.19 ± 0.48	58.09 ± 1.10
Kashin-DCT+K	16.57 ± 0.04	43.60 ± 0.18	50.00 ± 0.11	70.09 ± 0.30	57.22 ± 1.24	17.20 ± 0.02	**41.08 ± 0.04**	**40.63 ± 0.20**	70.31 ± 0.08	59.17 ± 0.80
Kashin-DCT+H	16.49 ± 0.07	**43.40 ± 0.27**	50.20 ± 0.07	70.42 ± 0.14	56.64 ± 0.92	**17.19 ± 0.02**	41.12 ± 0.02	40.59 ± 0.12	70.68 ± 0.28	58.32 ± 0.13

Method	Llama-2-7B					Llama-2-13B				
	Wiki-2 ↓	C4 ↓	Hella ↑	PiQA ↑	Wino ↑	Wiki-2 ↓	C4 ↓	Hella ↑	PiQA ↑	Wino ↑
FP16	9.20	19.40	76.14	78.73	69.46	8.11	17.59	79.64	80.36	72.53
RTN	10.56	22.78	74.43	78.56	68.75	8.74	18.71	78.80	79.65	70.88
OPTQ	9.80 ± 0.01	21.30 ± 0.06	74.79 ± 0.13	77.82 ± 0.27	68.25 ± 0.51	8.48 ± 0.00	18.59 ± 0.08	78.27 ± 0.34	80.11 ± 0.28	71.64 ± 0.59
QuIP	48.30† ± 29.84	133.00† ± 83.57	48.10† ± 9.88	68.41† ± 3.99	57.06† ± 3.42	8.64 ± 0.03	18.80 ± 0.14	77.79 ± 0.30	79.78 ± 0.30	72.27 ± 0.25
QuIP-RG	20.19† ± 4.22	48.58† ± 12.63	61.22† ± 3.85	73.54† ± 1.33	64.56† ± 1.23	8.63 ± 0.07	18.84 ± 0.13	78.16 ± 0.20	79.52 ± 0.79	**73.22 ± 0.67**
QuIP #	**9.47 ± 0.01**	**20.16 ± 0.13**	74.98 ± 0.08	**78.49 ± 0.14**	69.32 ± 0.43	**8.31 ± 0.00**	**18.19 ± 0.01**	78.86 ± 0.09	80.16 ± 0.16	72.43 ± 0.18
Kashin-DCT+K	9.65 ± 0.01	20.45 ± 0.09	74.90 ± 0.15	78.09 ± 0.27	**69.43 ± 0.78**	8.38 ± 0.00	**18.19 ± 0.08**	78.78 ± 0.04	**80.54 ± 0.31**	72.74 ± 0.55
Kashin-DCT+H	9.60 ± 0.02	20.36 ± 0.03	**75.04 ± 0.16**	78.24 ± 0.11	69.27 ± 0.87	8.37 ± 0.01	**18.19 ± 0.01**	**78.90 ± 0.23**	80.12 ± 0.17	72.53 ± 0.21

Table 2: Perplexity on WikiText-2 and C4 for activation-aware baselines and Kashin-DCT at 4-bit per-output-channel weight quantization. Each cell is mean ± sample standard deviation over three calibration seeds; lower is better. Best results per column are in **bold**, and second-best results are underlined. All evaluations use `lm-evaluation-harness` with maximum sequence length 2048.

Method	Pythia-1.4B		OPT-1.3B		Llama-2-7B		Llama-2-13B		Llama-3-8B	
	Wiki-2 ↓	C4 ↓	Wiki-2 ↓	C4 ↓	Wiki-2 ↓	C4 ↓	Wiki-2 ↓	C4 ↓	Wiki-2 ↓	C4 ↓
AWQ	17.17 ± 0.02	45.58 ± 0.08	17.78 ± 0.01	42.47 ± 0.07	9.90 ± 0.01	21.24 ± 0.33	8.47 ± 0.01	18.69 ± 0.04	9.15 ± 0.01	20.95 ± 0.07
OmniQuant	**16.27 ± 0.04**	**43.01 ± 0.13**	17.41 ± 0.01	41.44 ± 0.03	9.86 ± 0.01	21.70 ± 0.08	8.46 ± 0.02	18.98 ± 0.08	9.42 ± 0.01	21.98 ± 0.04
Kashin-DCT+K	16.57 ± 0.04	43.60 ± 0.18	17.20 ± 0.02	**41.08 ± 0.04**	9.65 ± 0.01	20.45 ± 0.09	8.38 ± 0.00	**18.19 ± 0.08**	8.83 ± 0.11	20.61 ± 0.28
Kashin-DCT+H	16.49 ± 0.07	43.40 ± 0.27	**17.19 ± 0.02**	41.12 ± 0.02	**9.60 ± 0.02**	**20.36 ± 0.03**	**8.37 ± 0.01**	**18.19 ± 0.01**	**8.76 ± 0.00**	**20.37 ± 0.01**

layer, and OPTQ itself degrades to $\approx$ 380 Wiki-2 PPL, while Kashin-DCT remains numerically stable and stays within $\sim$ 0.3 Wiki-2 PPL of the FP16 reference of 8.63 (Table 4; full diagnostic in Appendix A). Across the four stress configurations we encountered — elevated variance on Llama-2-7B and Pythia-1.4B, catastrophic divergence on Pythia-6.9B, and NaN abort on Mistral-7B — Kashin-DCT is the only pipeline that remains numerically stable on every layer, which we read as evidence that the bounded-ℓ_∞ factor decomposition with an adaptive per-column codebook is structurally more robust to weight-distribution outliers than fixed-grid rounding driven through an ill-conditioned Hessian inverse.

6 Conclusion

We revisited Kashin-decomposition quantization for LLMs and resolved the three weaknesses that had limited its prior application: weak convergence under matrix reformulation, the $\mathcal{O}(N^2)$ cost of an explicit random orthogonal transform, and slow multi-restart k-means clustering. The proposed Greedy Algorithm with Alternating Updates retains the dictionary $Q_N \cup U_{\mathcal{E},\Phi} Q_N$ but fixes the order of u- and $\hat{v}$-updates in blocks of four, which places the dominant updates of each factor at known centroid locations $\pm c_1 \pm c_2$ and produces the empirically four-peaked distributions that 2-bit clustering exploits; the analysis in Section 4 establishes geometric residual decay at per-block rate $(1-\beta(N))^{1/2}$ with $\beta(N) = \alpha^2(N)/36$, and an ℓ_∞ bound of order $K^4(\log N)^2(\log\log N)^{12}/\sqrt{N}$, i.e. within polylogarithmic factors of the unconstrained scheme of Kashin et al. [2025]. Replacing the random orthogonal Q with a sign-randomized DCT removes the dense $N \times N$ matrix entirely and brings the per-iteration cost to $\mathcal{O}(N \log N)$, while closed-form initialization of cluster centers from the residual norms removes the k-means multi-restart outer loop, reducing k-means clustering

wall-clock time. Composed with OPTQ-style error compensation and QuIP-style incoherence preprocessing, the resulting JAX pipeline is competitive with OPTQ, QuIP, QuIP-RG, and a fine-tuning- and vector-quantization-free variant of QuIP# at 4-bit per channel on OPT-1.3B, Llama-2-7B/13B, and Pythia-1.4B/6.9B. It is also competitive with AWQ and OmniQuant on WikiText-2 and C4 perplexity for OPT-1.3B, Llama-2-7B/13B, Llama-3-8B, and Pythia-1.4B (Table 2), while admitting an inference-time decomposition into two 2-bit factor codes per channel that is structurally well-suited to native-2-bit hardware (Appendix D). The favorable scaling of Kashin-DCT+H from 7B to 13B (Appendix E) further suggests that the relative cost of the proposed pipeline becomes more attractive as model size grows, even before any kernel-level optimization is applied. A second, less expected property emerges from the stress tests: on configurations where the QuIP family diverges into four-digit perplexities (Pythia-6.9B) or aborts with NaNs in LDL back-substitution (Mistral-7B v0.1), and where OPTQ itself silently saturates to ≈ 380 Wiki-2 PPL on the same layer, Kashin-DCT+H is the only pipeline that produces robust results (Appendix A); the bounded-ℓ_∞ factor decomposition with an adaptive per-column codebook is structurally more robust to weight-distribution outliers than fixed-grid rounding driven through an ill-conditioned Hessian inverse.

Limitations. We do not directly compare with rotation-based methods [Ashkboos et al., 2024, Liu et al., 2025]. That said, rotation-based methods are orthogonal to our weight decomposition rather than competing with it: the rotation is applied to the weight matrix prior to factorization, and our **Kashin-DCT+K** and **Kashin-DCT+H** variants are already exactly this composition with Kronecker and randomized-Hadamard rotations, so substituting any other orthogonal preprocessing (e.g. the learned rotations of Liu et al. [2025] or QuaRot's residual-stream rotation [Ashkboos et al., 2024]) leaves the rest of the pipeline unchanged. We also have not implemented the fused native-2-bit kernel analyzed in Appendix D; consequently, its expected efficiency benefits remain theoretical, and we do not report measured inference latency or throughput. Both directions are left for future work.

Future work. Two directions stand out. First, implementing the fused 2-bit GEMM with shared-ε DCT kernel described in Appendix D and benchmarking end-to-end latency against a tuned 4-bit baseline such as Marlin [Frantar et al., 2025] would convert the asymptotic bandwidth and arithmetic advantages of the decomposed form into a measured serving benefit, particularly on accelerators with native 2-bit tensor cores. Second, extending the decomposition beyond per-channel 4-bit — to sub-4-bit budgets via vector-quantization codebooks, or to weight-and-activation quantization by composing with learned rotations [Liu et al., 2025].

Acknowledgments and Disclosure of Funding

The work of Boris Kashin was carried out at Lomonosov Moscow State University with support from the Russian Science Foundation (Grant No. 24-11-00114).

References

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=OUIFPHEgJU.

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

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=ksVGCOlOEba.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xrk9g5vcXR.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: even better llm quantization with hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024a.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

Mart van Baalen, Andrey Kuzmin, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. GPTVQ: The blessing of dimensionality for LLM quantization. *arXiv preprint arXiv:2402.15319*, 2024.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De Sa. QTIP: Quantization with trellises and incoherence processing. In *Advances in Neural Information Processing Systems*, 2024b.

Daniil Merkulov, Daria Cherniuk, Alexander Rudikov, Ivan Oseledets, Ekaterina Muravleva, Aleksandr Mikhalev, and Boris Kashin. Quantization of large language models with an overdetermined basis. In Negar Kiyavash and Joris M. Mooij, editors, *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, volume 244 of *Proceedings of Machine Learning Research*, pages 2527–2536. PMLR, 15–19 Jul 2024. URL `https://proceedings.mlr.press/v244/merkulov24a.html`.

Boris Sergeevich Kashin. Diameters of some finite-dimensional sets and classes of smooth functions. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, 41(2):334–351, 1977.

Boris Kashin, Ivan Oseledets, and Alexander Rudikov. Accelerated algorithm for splitting a vector into two vectors with small uniform norm. *Matematicheskie Zametki*, 118(3):434–442, 2025.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, 2023.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, 2024a.

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 *The Twelfth International Conference on Learning Representations*, 2024.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

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.

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 *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=ogO6DGE6FZ`.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. DuQuant: Distributing outliers via dual transformation makes stronger quantized LLMs. In *Advances in Neural Information Processing Systems*, 2024b.

Vladimir Temlyakov. *Greedy approximation*, volume 20. Cambridge University Press, 2011.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *ArXiv*, abs/2106.08295, 2021. URL `https://api.semanticscholar.org/CorpusID:235435934`.

Olivier Guédon, Shahar Mendelson, Alain Pajor, and Nicole Tomczak-Jaegermann. Majorizing measures and proportional subsets of bounded orthonormal systems. *Revista Matematica Iberoamericana*, 24:1075–1095, 02 2008. doi: 10.4171/RMI/567.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open pre-trained transformer language models, 2022.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B, 2023.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, 2023.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

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

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*, 2019.

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*, 2020.

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.

Elias Frantar, Roberto L. Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. MARLIN: Mixed-precision auto-regressive parallel inference on large language models. In *Proceedings of the 30th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP'25, 2025.

A Stress Tests on Pythia-6.9B and Mistral-7B

Pythia-6.9B. Table 3 reports the full numbers for the Pythia-6.9B stress-test referenced in Section 5. Calibration setup is identical to the main table (1000 WikiText-2 train samples, sequence length 2048, three random seeds; mean $\pm$ std reported). The QuIP family fails catastrophically on this model (perplexities in the hundreds-to-thousands), while Kashin-DCT+H stays within an order of magnitude of OPTQ.

Table 3: Pythia-6.9B at 4-bit per channel. WikiText-2 / C4 perplexity (lower is better), HellaSwag / PiQA / Winogrande accuracy (higher is better). Best quantization result in **bold**, second best underlined (FP16/RTN excluded).

Method	Wiki-2 ↓	C4 ↓	Hella ↑	PiQA ↑	Wino ↑
FP16	11.41	30.11	63.87	76.44	61.40
RTN	17.55	44.59	58.86	73.01	59.12
OPTQ	**12.02 ± 0.00**	**32.06 ± 0.02**	**62.95 ± 0.32**	**75.90 ± 0.22**	**60.72 ± 0.12**
QuIP	2486.39 ± 569.00	8605.29 ± 2456.10	34.01 ± 1.56	54.93 ± 0.63	53.04 ± 0.93
QuIP-RG	2239.41 ± 778.10	7323.57 ± 2833.40	34.72 ± 0.97	55.50 ± 0.36	52.01 ± 1.12
QuIP #	324.93 ± 47.63	799.77 ± 96.96	41.13 ± 0.48	58.89 ± 0.17	57.01 ± 1.15
Kashin-DCT+K	25.94 ± 0.74	61.98 ± 0.97	57.64 ± 0.41	71.87 ± 0.24	60.33 ± 0.28
Kashin-DCT+H	20.64 ± 1.63	49.91 ± 3.13	58.66 ± 0.47	72.72 ± 0.19	59.98 ± 0.08

Mistral-7B v0.1. Table 4 reports per-method results on Mistral-7B v0.1 at 4-bit per channel under the same calibration setup as Table 3 (Kashin rows averaged over two seeds, GPTQ over three). The QuIP family fails to produce any output on Mistral, for the reason explained in the next paragraph. GPTQ runs to completion but degrades catastrophically (Wiki-2 PPL ≈ 380, more than $40\times$ above the FP16 reference of 8.63), while Kashin-DCT+K and Kashin-DCT+H remain numerically stable at 0.32 and 0.29 Wiki-2 PPL above the FP16 baseline, respectively.

Table 4: Mistral-7B v0.1 at 4-bit per channel. WikiText-2 / C4 perplexity, HellaSwag / PiQA / Winogrande accuracy. "aborts" = produces NaNs and fails to complete; see paragraph below. Best non-aborting quantization result per column in **bold**, second best underlined.

Method	Wiki-2 ↓	C4 ↓	Hella ↑	PiQA ↑	Wino ↑
FP16	8.63	21.42	81.23	82.75	75.14
GPTQ	379.73 ± 15.71	1948.81 ± 542.40	31.68 ± 0.86	67.17 ± 1.34	53.28 ± 0.85
QuIP / QuIP-RG / QuIP#	aborts (NaN in LDL decomposition; see below)				
Kashin-DCT+K	8.95 ± 0.01	22.26 ± 0.01	79.99 ± 0.15	81.59 ± 0.04	**74.55 ± 0.06**
Kashin-DCT+H	**8.92 ± 0.01**	**22.21 ± 0.01**	**80.09 ± 0.01**	**82.10 ± 0.23**	73.92 ± 0.95

Why all OPTQ-family methods fail on Mistral. Both failures share a root cause: the SwiGLU input to Mistral's `mlp.down_proj` produces extreme channel-wise outliers, leaving the 14336×14336 calibration Hessian $H = X^\top X$ ill-conditioned with several eigenvalues orders of magnitude below the largest; randomized rotations do not change the spectrum, so the conditioning survives the OPTQ-default 1% diagonal damping. *QuIP (LDLQ)* aborts: the Cholesky factor L has near-zero diagonals, LDLQ's unit-diagonal normalization $L \leftarrow L\,\mathrm{diag}(L)^{-1}$ amplifies off-diagonals, and the back-substitution overflows to NaN at `mlp.down_proj` on both v0.1 and v0.3 (all four had/kron × ldlq/ldlqRG variants). *OPTQ* runs to completion but silently destroys the model: the residual update

$$W_{:,i+1:} \;\leftarrow\; W_{:,i+1:} \;-\; \frac{W_{:,i} - Q_{:,i}}{[H^{-1}]_{ii}}\,[H^{-1}]_{i,i+1:} \tag{7}$$

uses the same ill-conditioned H^{-1}, swinging the yet-to-be-quantized columns far outside their pre-calibrated per-channel grid; the subsequent round-to-grid step saturates them at the endpoints (`--act-order` mitigates this only marginally). *Kashin-DCT* remains robust: the analytic centroids $\pm c_1 \pm c_2$ scale with the actual residual norm $\|r_k\|_1/N$, so the codebook follows the inflated magnitude, and the bounded-ℓ_∞ factor decomposition keeps k-means well-conditioned on the same calibration data. The same mechanism is the source of stability we observe in Table 1 on the Llama-2-7B columns where LDLQ+Kron variance blows up (the daggered † cells), and in Table 3 on Pythia-6.9B where all three QuIP variants diverge into the hundreds-to-thousands PPL range while Kashin-DCT+H stays within an order of magnitude of the OPTQ baseline.

B Original Kashin Greedy Algorithm

For completeness, we restate the original Kashin vector-decomposition greedy algorithm, on which the matrix reformulation of Merkulov et al. [2024] and our partitioned variant in Section 4 build.

Algorithm 2 Vector Decomposition Greedy Algorithm

Input: Vector $x \in \mathbb{R}^n$, Orthogonal matrix Q, Tolerance $\varepsilon > 0$
Output: Vectors $u, \hat{v} \in \mathbb{R}^n$ such that $x \approx u + v = u + Q^T\hat{v}$, and both u and $hatv$ have small infinity norm.
Initialize $u \leftarrow 0^n$, $\hat{v} \leftarrow 0^n$
Define projection $\pi_x(y) := \dfrac{x^\top y}{\|y\|_2^2} \cdot y$
while $\|x\| \geq \varepsilon$ **do**
  if $\|x\|_1 > \|Qx\|_1$ **then**
    $\pi \leftarrow \pi_x(\mathrm{Sign}(x))$
    $u \leftarrow u + \pi$
    $x \leftarrow x - \pi$
  else
    $\pi \leftarrow \pi_x(\mathrm{Sign}(Qx))$
    $\hat{v} \leftarrow \hat{v} + \pi$
    $x \leftarrow x - Q^T\pi$
  end if
end while
Return: $x, u, \hat{v}$

C Ablation Studies

We ablate the three components introduced by our method: the structured DCT-based transform, the block-greedy schedule, and the closed-form k-means initialization.

C.1 DCT-based transform versus a dense random orthogonal transform

We decompose all 1,024 columns of `model.decoder.layers[0].fc1` from `facebook/opt-350m`, using 100 greedy iterations per column and no quantization. We compare the dense random orthogonal transform used in prior work, constructed by applying QR factorization to a Gaussian matrix, with our sign-randomized DCT operator P. Table 5 reports reconstruction quality, convergence speed measured as orders of magnitude of residual-norm reduction per 10 iterations, and wall-clock time per column.

Table 5: Transform ablation over all 1,024 columns of the first OPT-350m decoder-layer `fc1` matrix. Values are population mean $\pm$ population standard deviation across columns. Higher convergence speed is better; lower reconstruction error and time are better. Timing includes transform construction and decomposition but excludes reconstruction-history analysis.

Transform	Relative ℓ_2 error $\downarrow$	Orders / 10 iterations $\uparrow$	Time / column (ms) $\downarrow$	Relative time
Dense random orthogonal (QR)	$1.733934 \times 10^{-4} \pm 9.430639 \times 10^{-5}$	1.938669 ± 0.067113	39.330077 ± 0.224382	$6.625517\times$
Sign-randomized DCT (ours)	$\mathbf{2.571530 \times 10^{-7} \pm 1.074186 \times 10^{-7}}$	$\mathbf{2.057676 \pm 0.028330}$	$\mathbf{5.936152 \pm 0.189360}$	$\mathbf{1.000000}\times$

The structured transform is approximately $6.6\times$ faster per column and reaches nearly three orders of magnitude lower reconstruction error at the same iteration budget. Thus, the speedup does not trade off reconstruction accuracy.

C.2 Block-greedy schedule

Figure 2 serves as the ablation of the block-greedy schedule. The original, unpartitioned vector greedy algorithm and Algorithm 1 converge at nearly the same rate, but only the partitioned schedule produces four sharp symmetric peaks in both factors. The original schedule leaves the second factor unimodal, making its 2-bit clustering degenerate. Figure 3(b)–(c) shows the same distinction on a real model layer. Convergence alone therefore does not guarantee a quantizable factor structure; the partitioned schedule does.

C.3 Closed-form k-means initialization

We compare clustering of the factors u and $\hat{v}$ with and without the closed-form $\pm c_1 \pm c_2$ initialization across all columns of the same OPT-350m `fc1` layer. The proposed configuration uses one k-means run initialized at the analytic peak locations, whereas the prior-work configuration uses random initialization with ten restarts.

Table 6: Initialization ablation over all 1,024 columns of the first OPT-350m decoder-layer `fc1` matrix. Full-column relative ℓ_2 error is reported as population mean $\pm$ population standard deviation across columns; lower is better.

Clustering configuration	Relative ℓ_2 error $\downarrow$
Closed-form initialization, one run (ours)	$\mathbf{0.120887 \pm 0.002272}$
Random initialization, ten restarts	0.137225 ± 0.005733

The closed-form initialization achieves lower quantization error with a single k-means run than random initialization achieves with ten restarts.

D Inference

The Kashin representation $w = u + P\hat{v}$ stores each layer as two 2-bit factor matrices $U, \hat{V} \in \mathbb{R}^{N \times M}$ together with the per-layer sign mask $\varepsilon \in \{\pm 1\}^N$ that defines the orthogonal operator $P = \mathrm{IDCT} \circ T_\varepsilon \circ \mathrm{DCT}$. Because ε is shared across all columns of a layer (generated once from a fixed seed), a single P acts on every column of $\hat{V}$. With the orthonormal DCT-II[3], P is a symmetric involution ($P^\top = P$ and $P^2 = I$, since T_ε is diagonal

[3]Throughout this paper we use the orthonormal DCT-II ($\mathrm{IDCT} = \mathrm{DCT}^\top$); the unscaled type-II DCT does not satisfy this identity and the involution $P^2 = I$ below would not hold.

with $\varepsilon_i^2 = 1$). Associativity of the matmul then gives the folding identity

$$XW \;=\; X\big(U + P\hat{V}\big) \;=\; XU + (XP)\hat{V} \;=\; \big[\, X \;\big|\; XP \,\big] \begin{bmatrix} U \\ \hat{V} \end{bmatrix}, \tag{8}$$

where XP is computed by applying the same FFT-fast 'iDCT $\circ$ sign-mask $\circ$ DCT' pipeline used for $P\hat{v}$ during quantization, just to each row of X rather than to a column of $\hat{V}$.

Compute. The right path through (8) depends on hardware. *Without* native 2-bit GEMM support, it is more efficient to reconstitute the dense weight $W = U + P\hat{V}$ once at load time and run a standard fp16 (or 4-bit-dequant) matmul XW. *With* native 2-bit GEMM (e.g. Hopper-class FP4/INT2 tensor cores), the fused form of (8) is preferable: keeping U and $\hat{V}$ in their 2-bit storage and computing $XU + (XP)\hat{V}$ as a single $2N$-K-dimension 2-bit GEMM avoids the fp16 expansion of the weight body and preserves the on-chip footprint advantage of 2-bit codes. In both regimes the DCT cost is $\mathcal{O}(KN \log N)$ for $K = (\text{batch}) \cdot (\text{seq len})$ activation rows, several orders of magnitude below the $\mathcal{O}(KNM)$ matmul.

Native 2-bit hardware. On accelerators with native 2-bit tensor cores, the decomposed form gains two kernel-level advantages over a standard 4-bit per-channel kernel: (i) codebook dequantization maps cleanly to the 2-bit addressing unit – the lookup $c[\,\text{code}\,]$ replaces $(\text{code} - z_p) \cdot s$ operation of uniform quantization; (ii) each 2-bit code is the natural register-packing and shared-memory granule, so the centroid table loads contiguously without the cross-lane shuffles a 4-bit dequant with zero-point typically requires.

Memory bandwidth. At 4-bit per channel, each layer stores $4NM$ weight bits ($2NM$ for each of U and $\hat{V}$). Symmetric centroids about zero in both factors permit the per-column metadata to be reduced to two signed magnitudes per factor ($4M$ fp16 values per layer), plus a layer-shared sign vector $\varepsilon \in \{\pm 1\}^N$, which can be recovered from the seed rather than stored. The effective bits-per-weight is therefore $(4NM + 64M)/(NM) = 4 + 64/N$; for typical LLM hidden dimensions $N \in [4096, 8192]$ this metadata adds at most $64/4096 \approx 0.016$ bits per weight on top of the 4-bit weight body. By comparison, standard 4-bit per-channel quantization stores one fp16 scale per column and (optionally) one fp16 zero-point, yielding $4 + 16/N$ bits per weight without zero-point and $4 + 32/N$ with it – equivalently, an additional 0.004 and 0.008 bits per weight at $N = 4096$.

E Quantization Runtime

Table 7 reports the end-to-end wall-clock time required to quantize Llama-2-7B and Llama-2-13B at 4 bits per channel for the three methods we ran on identical hardware: OPTQ, the fine-tuning- and vector-quantization-free variant of QuIP# (LDLQ-RG with randomized Hadamard incoherence preprocessing), and our Kashin-DCT+H pipeline. All measurements are taken on a single NVIDIA H100 GPU; mean $\pm$ standard deviation is computed over three random seeds. The reported time does not include Hessian estimation.

Table 7: Quantization wall-clock time at 4 bits per channel. Single NVIDIA H100 GPU; mean $\pm$ std over three random seeds.

Method	Llama-2-7B (s) $\downarrow$	Llama-2-13B (s) $\downarrow$
OPTQ	229.9 ± 2.2	437.1 ± 2.1
QuIP# (LDLQ-RG + Hadamard)	763.0 ± 5.3	1298.0 ± 6.1
Kashin-DCT+H	1313.6 ± 26.7	1544.9 ± 10.8

Scaling 7B $\rightarrow$ 13B. Going from Llama-2-7B to Llama-2-13B (a $\sim 1.9\times$ parameter increase), OPTQ and QuIP# wall-clock times scale roughly proportionally – $1.90\times$ ($229.9 \rightarrow 437.1$ s) and $1.70\times$ ($763.0 \rightarrow 1298.0$ s), respectively – whereas Kashin-DCT+H grows only $1.18\times$ ($1313.6 \rightarrow 1544.9$ s). Concretely, doubling the model roughly doubles the OPTQ and QuIP# quantization budget but adds only $\sim 18\%$ to Kashin-DCT+H, so the relative cost of the proposed method becomes more favorable as model size grows.

F Proof of Proposition 1

The dictionary $\mathcal{S}$ is partitioned into two halves:

$$\mathcal{S} = Q_N \bigcup U_{\mathcal{E},\Phi} Q_N.$$

The *partitioned greedy algorithm* alternates between these halves: within each block, the first two steps use vectors from Q_N, and the next two steps use vectors from $U_{\mathcal{E},\Phi} Q_N$. To establish the convergence rate, it suffices to analyze the error reduction over a single block.

Fix one such block and denote by ρ_0 the residual at its start; thus $\rho_0 = x$ and $\|\rho_0\|_2 \leqslant A$. For $j \in \{1,\,2,\,3,\,4\}$, let ρ_j be the residual after the j-th step inside this block. In the notation of Proposition 1 we have $r_1 = \rho_4$.

By (3), for ρ_0 at least one of the following conditions holds:

$$\text{(i)} \quad \|\rho_0\|_1 \;\geqslant\; \frac{R(N)}{K\sqrt{2}}\,A, \tag{9}$$

$$\text{(ii)} \quad \|U_{\mathcal{E},\Phi}\,\rho_0\|_1 \;\geqslant\; \frac{R(N)}{K\sqrt{2}}\,A. \tag{10}$$

We first record the standard estimate used in Kashin et al. [2025]. Suppose $\|y\|_1 \;\geqslant\; \frac{R(N)}{K\sqrt{2}}\,\|y\|_2$. For any $w \in Q_N$ we have $\|w\|_2^2 = N$, and the maximum $\max_{w\in Q_N} |\langle y, w\rangle| = \|y\|_1$ is attained by $w_j^* = \operatorname{sign}(y_j)$. Hence the greedy step satisfies

$$\big\|y - \lambda^* w^*\big\|_2^2 = \|y\|_2^2 - \frac{\|y\|_1^2}{N} \;\leqslant\; \|y\|_2^2\Big(1 - \frac{R^2(N)}{2K^2N}\Big) = \|y\|_2^2\,(1-\alpha(N)). \tag{11}$$

Since $U_{\mathcal{E},\Phi}^2 = I$ (as $\mathcal{F}_\Phi$ is orthogonal transformation,$T_\mathcal{E}$ is diagonal, $T_\mathcal{E}^2 = I$), we have $\langle y,\, U_{\mathcal{E},\Phi} q\rangle = \langle U_{\mathcal{E},\Phi}\, y,\, q\rangle$ for any $q \in Q_N$, so an identical argument applies to $U_{\mathcal{E},\Phi} Q_N$ when condition (10) holds.

Case 1: condition (9) **holds.** By (11), the first greedy step (from Q_N) gives $\|\rho_1\|_2 \leqslant A\sqrt{1-\alpha(N)}$. The remaining three steps are non-expansive (each is an orthogonal projection onto a one-dimensional subspace), so

$$\|r_1\|_2 \;=\; \|\rho_4\|_2 \;\leqslant\; \|\rho_1\|_2 \;\leqslant\; A\sqrt{1-\alpha(N)} \;\leqslant\; A\sqrt{1-\beta(N)}.$$

Case 2: condition (9) **fails.** By (3), condition (10) must then hold.

Sub-case 2a: $\|\rho_2\|_2 < A\sqrt{1-\beta(N)}$. The third and fourth steps are non-expansive, hence $\|r_1\|_2 = \|\rho_4\|_2 \leqslant \|\rho_2\|_2 \leqslant A\sqrt{1-\beta(N)}$, and (5) holds.

Sub-case 2b: $\|\rho_2\|_2 \geqslant A\sqrt{1-\beta(N)}$. By the property of the pure greedy algorithm ($\rho_j \perp w_j$ at each step j),

$$\|x\|_2^2 = \|\rho_2\|_2^2 + \|\lambda_1 w_1\|_2^2 + \|\lambda_2 w_2\|_2^2.$$

Combining with $\|x\|_2 \leqslant A$ and $\|\rho_2\|_2 \geqslant A\sqrt{1-\beta(N)}$ yields

$$\|\lambda_1 w_1\|_2^2 + \|\lambda_2 w_2\|_2^2 \;\leqslant\; A^2\,\beta(N),$$

so the triangle inequality gives

$$\|\lambda_1 w_1 + \lambda_2 w_2\|_2 \;\leqslant\; \|\lambda_1 w_1\|_2 + \|\lambda_2 w_2\|_2 \;\leqslant\; A\sqrt{2\,\beta(N)}.$$

Let $\lambda^*\,w^*$ be the best approximation of x from $U_{\mathcal{E},\Phi} Q_N$. Since condition (10) holds, the one-step estimate gives $\|x - \lambda^* w^*\|_2 \leqslant A\sqrt{1-\alpha(N)}$. Because the actual greedy choice at the third step is at least as good as $\lambda^*\,w^*$, the triangle inequality yields

$$\begin{aligned}
\|\rho_3\|_2 \;&\leqslant\; \|\rho_2 - \lambda^* w^*\|_2 \;\leqslant\; \|x - \lambda^* w^*\|_2 + \|\lambda_1 w_1 + \lambda_2 w_2\|_2 \\
&\leqslant\; A\sqrt{1-\alpha(N)} \;+\; A\sqrt{2\,\beta(N)} \\
&\leqslant\; A\big(1 - \tfrac{\alpha(N)}{2} + \tfrac{\alpha(N)\sqrt{2}}{6}\big) \;\leqslant\; A\big(1 - \tfrac{\alpha(N)}{4}\big) \;\leqslant\; A\sqrt{1-\beta(N)}.
\end{aligned} \tag{12}$$

The fourth step is non-expansive, hence

$$\|r_1\|_2 = \|\rho_4\|_2 \leqslant \|\rho_3\|_2 \leqslant A\sqrt{1-\beta(N)} \tag{13}$$

In all cases (5) is established, completing the proof.

G Proof of Theorem 2

We prove

$$\|r_k\|_2 \;=\; \|x - u_k - v_k\|_2 \;\leqslant\; \big(1-\beta(N)\big)^{k/2} \tag{14}$$

by induction on k. The case $k = 0$ is immediate: $\|r_0\|_2 = \|x\|_2 \leqslant 1 = (1-\beta(N))^0$.

Assume (14) holds for some $k \geqslant 0$ and consider the $(k+1)$-th block, which starts from $\rho_0 = r_k$ and produces $r_{k+1} = \rho_4$. Setting $A = (1-\beta(N))^{k/2}$ in Proposition 1, we obtain

$$\|r_{k+1}\|_2 \;\leqslant\; A\sqrt{1-\beta(N)} \;=\; \big(1-\beta(N)\big)^{(k+1)/2},$$

which closes the induction and proves the first line of (6).

Recall that within the j-th block the algorithm performs four atomic steps: two from Q_N (contributing to u) and two from $U_{\mathcal{E},\Phi}\,Q_N$ (contributing to v). Let $w_j^{(1)}, w_j^{(2)} \in Q_N$ and $w_j^{(3)}, w_j^{(4)} \in U_{\mathcal{E},\Phi} Q_N$ denote the dictionary elements chosen in block j, with corresponding coefficients $\lambda_j^{(1)}, \dots, \lambda_j^{(4)}$. By construction,

$$u_k \;=\; \sum_{j=1}^{k} \sum_{\ell\in\{1,2\}} \lambda_j^{(\ell)}\, w_j^{(\ell)}, \quad v_k \;=\; \sum_{j=1}^{k} \sum_{\ell\in\{3,4\}} \lambda_j^{(\ell)}\, w_j^{(\ell)}. \tag{15}$$

Coefficient bound. At every greedy step the chosen atom w satisfies $\|w\|_2^2 = N$ and the associated coefficient equals $\lambda = \langle \rho, w\rangle / N$, where ρ is the residual entering that step. Hence by Cauchy–Schwarz,

$$|\lambda_j^{(\ell)}| \;\leqslant\; \frac{\|\rho_{j,\ell-1}\|_2\,\|w\|_2}{\|w\|_2^2} \;=\; \frac{\|\rho_{j,\ell-1}\|_2}{\sqrt{N}}, \tag{16}$$

where $\rho_{j,\ell-1}$ is the residual just before step ℓ of block j. Since atomic steps are non-expansive, $\|\rho_{j,\ell-1}\|_2 \leqslant \|r_{j-1}\|_2$, and by part 1, $\|r_{j-1}\|_2 \leqslant (1-\beta(N))^{(j-1)/2}$. Therefore

$$|\lambda_j^{(\ell)}| \;\leqslant\; \frac{(1-\beta(N))^{(j-1)/2}}{\sqrt{N}}. \tag{17}$$

Summing the contributions. Every atom $w \in Q_N$ has $\|w\|_\infty = 1$. By (15) and (17),

$$\begin{aligned} \|u_k\|_\infty \;\leqslant\; \sum_{j=1}^{k} \sum_{\ell\in\{1,2\}} |\lambda_j^{(\ell)}| \;&\leqslant\; \frac{2}{\sqrt{N}} \sum_{j=1}^{k} \big(1-\beta(N)\big)^{(j-1)/2} \;\leqslant \\ &\leqslant\; \frac{2}{\sqrt{N}} \cdot \frac{1}{1-\sqrt{1-\beta(N)}} \;\leqslant\; \frac{4}{\beta(N)\sqrt{N}}. \end{aligned} \tag{18}$$

The same bound holds for $\|U_{\mathcal{E},\Phi}\, v_k\|_\infty$ since $U_{\mathcal{E},\Phi}^2 = I$ implies that the action of $U_{\mathcal{E},\Phi}$ on the second sum in (15) produces atoms in Q_N with the same ℓ_∞-norm bound.

Combining (15) with the fact that $\|w\|_\infty = 1$ for every $w \in \mathcal{S}$ and substituting $\beta(N) = \alpha^2(N)/36 = R^4(N)/(144\,K^4\,N^2) = 1/(144\,c_3^4\,K^4\,(\log N)^2\,(\log\log N)^{12})$, we conclude

$$\max\Big(\|u_k\|_\infty,\;\; \|U_{\mathcal{E},\Phi}\, v_k\|_\infty\Big) \;\leqslant\; \sum_{j,\ell} |\lambda_j^{(\ell)}| \;\leqslant\; \frac{4}{\beta(N)\sqrt{N}} \;=\; \frac{c_4\,K^4\,(\log N)^2\,(\log\log N)^{12}}{\sqrt{N}},$$

where $c_4 = 576\,c_3^4$ is an absolute constant. This is exactly the second line of (6).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the abstract and introduction section we highlight our contributions that we discuss in Section 4 and demonstrate in Section 5

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, In Section 6, paragraph "Limitations and future work", we discuss the limitations of the current work

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Appendix F contains proofs of both Proposition and Theorem, proposed in Section 4. Additionally, the end of the Section provides a sketch of both proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, we specify the hyperparameters, datasets, models and hardware. As well as number of random seeds. Moreover, we provide the full source code of our experiments, including the methods that we compare with.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The supplementary material provides the code needed to run all of the experiments presented in the paper, including the methods we compare to. The instructions are listed in the several READMEs provided in the Supplementary.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The Experiments section (5) specifies on hyperparameters needed to reproduce the experiments. Additionally, in the Supplementary, we provide the bash scripts that specify all the arguments we used when running experiments presented in the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The Results in the Experiments Section (5) all specify standard deviation, where it is applicable (FP16 and RTN results do not vary). The section also specifies where the variability comes from.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Yes, both the Experiments section (5) and the Appendix E contain details on the hardware. Appendix E provides estimations on runtime. Appendix D provides details on memory footprint and inference.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The paper and the Supplementary material were anonimized.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The Introduction (Section 1) notes the positive impact: lower per-query energy use and a smaller carbon footprint. Quantization is a capability-preserving compression technique — it does not unlock new behaviors or categories of misuse beyond those of the unquantized model — so any negative societal impact is inherited from the choice of underlying LLM rather than introduced by this work.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Not applicable to the present research, since it is concerned with quantization of open LLMs.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The datasets (wikitext, c4), frameworks (lm-evaluation-harness), and implementations (optq, quip, quip#) were properly cited in the paper.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: Not applicable to the present research, since it is concerned with quantization of open LLMs.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: Not applicable to the present research, since it is concerned with quantization of open LLMs.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Not applicable to the present research, since it is concerned with quantization of open LLMs.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.