

3% Is All You Need: Breaking TurboQuant’s Compression Limit via Spectral Structure

Ashwin Gopinath

Sentra, 235 2nd Street, San Francisco, CA 94105

*Department of Mechanical Engineering, MIT,
77 Massachusetts Avenue, Cambridge, MA 02139*

ashwin@sentra.app agopi@mit.edu

April 2026

Abstract

TURBOQUANT [Zandieh et al., 2025] established that data-oblivious KV cache compression—requiring zero calibration—can achieve within $2.7\times$ of the information-theoretic optimum. The result, from Google Research, drew immediate attention: multiple open-source implementations appeared within days—and inference teams began integrating TURBOQUANT into production serving stacks. We ask a simple question: what if 15 seconds of setup is worth it?

We present SPECTRALQUANT, a data-aware algorithm that exploits a universal structural property: across four transformer models (Qwen 2.5 at 1.5B, 7B, 14B; Llama 3.1-8B), the effective dimensionality of KV cache representations is $d_{\text{eff}} \approx 4$ out of 128 dimensions—a 97% spectral gap.

By rotating representations into signal-aligned coordinates and selectively discarding error correction on the 97% of dimensions that carry noise, SPECTRALQUANT simultaneously reduces memory use and improves reconstruction fidelity. The central finding is counterintuitive: *removing* a form of error correction on noise dimensions actually *improves* quality, because error correction on random noise injects variance without reducing bias.

On Qwen 2.5-14B, SPECTRALQUANT achieves cosine similarity **0.9485** versus TURBOQUANT’s **0.9226** (+2.59 percentage points, pp) at compression ratio $5.95\times$ versus $5.02\times$ —better quality *and* better compression at the same bit budget. Results replicate across all four models and all five independent random-seed trials, with calibration requiring 15 seconds on a single GPU, performed once before deployment. Perplexity is identical to uncompressed inference (9.51 for all methods), and all methods achieve perfect needle-in-a-haystack retrieval at context lengths up to 8,192 tokens.

The key insight—that uniform error correction on spectrally concentrated representations injects more noise than it removes—has implications for any compression pipeline applied to structured learned representations.

1 Introduction

In early 2025, Google Research published TURBOQUANT, a KV cache compression algorithm with a striking theoretical guarantee: provably within $2.7\times$ of the information-theoretic optimum, with zero calibration. The result drew immediate attention—multiple open-source implementations appeared within 48 hours, and inference teams began integrating TURBOQUANT into production serving stacks. The appeal was obvious: compress the KV cache by $5\times$ with no setup, no training data, no per-model tuning.

This paper asks a simple question: what if 15 seconds of setup is worth it?

The KV cache bottleneck. During inference, language models maintain the key-value (KV) cache: a growing record of key and value vectors for every token processed. At 8,192 tokens of context, Qwen 2.5-14B accumulates 320 MB of cache data per session—48 layers $\times$ 8 KV heads $\times$ 128 dimensions $\times$ 2 (key/value) $\times$ 8,192 tokens $\times$ 2 bytes. For a server running 100 concurrent sessions, that is 32 GB dedicated solely to remembering context, before model weights are counted. Memory, not compute, is the primary constraint on serving scale.

TurboQuant’s solution and its limit. TURBOQUANT [Zandieh et al., 2025] answers the compression question for methods that make no assumptions about the data. By applying a random rotation followed by uniform 3-bit quantization with Johnson-Lindenstrauss (QJL) error correction, TURBOQUANT achieves $5.02\times$ compression and establishes a data-oblivious bound: no method that treats dimensions identically can substantially improve on its guarantees. The bound is tight within the data-oblivious class. However, it says nothing about data-aware methods that spend a small amount of time examining the data first.

Our discovery. We find that the effective dimensionality of KV cache key vectors concentrates in $\approx 3\text{--}4\%$ of the head dimension—universally, across six models in four families and a $10\times$ range of parameter counts. On standard 128-dim heads, $d_{\text{eff}}^{\text{keys}} \approx 4$; on Gemma 2-9B’s 256-dim heads, $d_{\text{eff}}^{\text{keys}} \approx 8.2$ —the same 3.2% ratio. The spectral gap means that 96–97% of dimensions carry noise rather than signal. TURBOQUANT’s QJL error correction, applied uniformly to all dimensions, injects variance into those noise dimensions without reducing their bias—because there is no signal to recover. Removing QJL from noise dimensions yields +1.7 to +2.8 pp better cosine similarity at 18.6% better compression ratio, with SPECTRALQUANT faster than TURBOQUANT at all tested sequence lengths ($2.2\times$ per-step speedup at 512 tokens).

Contributions. This paper makes five contributions:

- **Empirical spectral characterization.** We measure $d_{\text{eff}}/d \approx 3\text{--}4\%$ across six models in four families (Qwen, Llama, Mistral, Gemma), including Gemma 2-9B with $d = 256$ (double the standard head dimension), and show this ratio is stable across data splits (CV = 3.9%).
- **SpectralQuant algorithm.** A three-part modification of TURBOQUANT: (1) replace random rotation with calibrated eigenvector rotation; (2) apply QJL error correction only to signal dimensions; (3) allocate bits non-uniformly.

- **Evaluation across eight axes.** Cosine similarity, perplexity, needle-in-a-haystack retrieval, long-context quality, distribution shift, latency, and cross-sequence-length latency crossover analysis.
- **KV spectral asymmetry.** Systematic measurement of $d_{\text{eff}}^{\text{keys}} \ll d_{\text{eff}}^{\text{values}}$ across five models, explaining why low-rank compression fails for values while SPECTRALQUANT succeeds.
- **Theoretical interpretation.** We explain the counterintuitive QJL-removal benefit via a bias-variance decomposition, showing that selective correction reduces overall mean squared error.

2 The Spectral Structure of Neural Representations

2.1 Empirical Finding and Theoretical Motivation

The central premise of SPECTRALQUANT is that KV cache vectors do not uniformly occupy their full 128-dimensional space. We begin with the empirical finding, then provide the theoretical motivation.

Box 1: What is the KV cache?

When a language model processes a sequence, each attention layer computes two vectors per token: a **key** (what that token is about) and a **value** (what information it contributes). These are cached so they need not be recomputed for every new token generated. The cache grows with context length: Qwen 2.5-14B accumulates 320 MB at 8,192 tokens for 100 concurrent users, that is 32 GB dedicated solely to remembering context—before model weights are counted. Compressing the KV cache is therefore the primary lever for increasing serving throughput.

We measured the participation ratio across four models. Table 1 shows the result: $d_{\text{eff}} \approx 4$ universally.

Table 1: **Spectral universality across six transformer models and four families.** $d_{\text{eff}}^{\text{keys}}$ is the mean participation ratio of key vectors across all layers and KV heads. The ratio d_{eff}/d is the fraction of the head dimension that carries signal. Despite spanning four architectural families, a $6\times$ range of parameter counts, and two head dimensions (128 and 256), d_{eff}/d clusters tightly at 3.1–3.4%—a 96–97% spectral gap in every case. This universality is the central empirical finding.

Model	Family	head_dim	$d_{\text{eff}}^{\text{keys}}$	κ	d_{eff}/d
Qwen 2.5-1.5B-Instruct	Qwen	128	3.95	1.247	3.1%
Qwen 2.5-7B-Instruct	Qwen	128	4.30	1.238	3.4%
Qwen 2.5-14B-Instruct	Qwen	128	4.17	1.220	3.3%
Llama 3.1-8B-Instruct	Llama	128	3.64	1.257	2.9%
Mistral 7B-Instruct	Mistral	128	4.18	1.222	3.3%
Gemma 2-9B-IT	Gemma	256	8.20	1.114	3.2%

The consistency is striking. On 128-dim heads, $d_{\text{eff}}^{\text{keys}}$ ranges from 3.64 (Llama) to 4.30 (Qwen 7B)—all within a 0.66-unit band. Gemma 2-9B uses 256-dim heads; its $d_{\text{eff}}^{\text{keys}} = 8.20$ is twice as large in absolute terms but the same 3.2% ratio. This proportionality confirms that the spectral gap is a property of the head architecture, not a fixed absolute number. The noise fraction—96–97% of dimensions—is universal.

Let $\{h_t\}_{t=1}^T \subset \mathbb{R}^d$ denote the sequence of key (or value) vectors accumulated in a single attention head during a forward pass, where $d = 128$ is the head dimension. Define the empirical covariance

$$\Sigma = \frac{1}{T} \sum_{t=1}^T h_t h_t^\top \in \mathbb{R}^{d \times d},$$

with eigendecomposition $\Sigma = U \Lambda U^\top$, where $\Lambda = \text{diag}(\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \geq 0)$.

Definition 2.1 (Participation Ratio / Effective Dimensionality). *The participation ratio (PR) of Σ is*

$$d_{\text{eff}} = \text{PR}(\Sigma) = \frac{\left(\sum_{i=1}^d \lambda_i\right)^2}{\sum_{i=1}^d \lambda_i^2}.$$

The participation ratio ranges from 1 (all variance in one direction) to d (variance spread uniformly). Prior work [Barman et al., 2026b,a] established that learned representations have bounded PR: information concentrates in few directions because training encourages representations to become geometrically structured. The *spectral gap ratio* κ quantifies how sharp this concentration is:

$$\kappa = \frac{\lambda_{\lceil d_{\text{eff}} \rceil}}{\lambda_{\lceil d_{\text{eff}} \rceil + 1}}.$$

When $\kappa \gg 1$, a natural cut-point separates signal from noise.

Box 3: What is the spectral gap?

Every set of vectors has a natural *importance ranking* of directions, given by **eigenvalues**—“importance scores” for each dimension. The **spectral gap** is the jump between the last important direction and the first unimportant one. When this jump is large, compression is straightforward: keep important directions at high fidelity, discard the rest.

For KV cache vectors in transformer models, the spectrum shows exactly this structure. The effective dimensionality $d_{\text{eff}} = \text{PR}(\Sigma)$ is ≈ 4 out of 128—only 4 directions carry measurable signal; the remaining 124 are dominated by noise and can be compressed aggressively with minimal loss.

Calibration stability. A natural concern is whether d_{eff} depends on the specific calibration data used. We test this by measuring d_{eff} on three independent 50-sample splits of the calibration set. The mean d_{eff} across splits is 3.93, 3.71, and 3.61; the coefficient of variation across heads is $\text{CV} = 3.9\%$. Only 1 of 56 heads shows $\text{CV} > 20\%$. The spectral structure is highly stable.

2.2 Related Work

With the spectral structure of KV cache representations established, we situate SPECTRALQUANT among prior methods that have approached KV cache compression from different angles.

KV cache quantization. KIVI [Liu et al., 2024] proposes asymmetric 2-bit quantization for KV caches, applying 2-bit precision to keys and 4-bit to values without calibration. KVQuant [Hooper et al., 2024] extends this to extreme contexts (up to 10M tokens) via per-channel and per-vector quantization with sensitivity analysis. GEAR [Feng et al., 2024] adds a low-rank error correction step on top of quantization, achieving near-lossless compression at $4\times$. H2O [Ge et al., 2024] takes a different approach, pruning KV cache entries rather than quantizing them, based on attention score magnitude.

Rotation-based methods. TURBOQUANT [Zandieh et al., 2025] establishes the data-oblivious bound via random rotation followed by uniform quantization and QJL error correction, achieving within $2.7\times$ of the information-theoretic optimum. TURBOQUANT’s theoretical guarantee is tight within its class: no data-oblivious method can substantially improve on it. No official implementation was released by the Google Research authors; community implementations appeared within 48 hours of publication, including Vangara [Vangara, 2026] and scos-lab [scos-lab, 2025]. QuIP [Chee et al., 2024] applies incoherence processing (random rotation) before quantization for weight matrices, showing that rotation reduces the dynamic range of values that must be quantized; SPECTRALQUANT applies the analogous insight to activation vectors but uses calibrated rather than random rotation.

Spectral and PCA-based methods. KVTC [NVIDIA Research, 2025] performs channel-wise decorrelation of KV vectors before quantization, achieving improved compression by reducing cross-channel correlations. ScaNN [Guo et al., 2020] uses PCA-based dimensionality reduction for efficient approximate nearest-neighbor search, a closely related problem to KV cache compression. Both methods exploit spectral structure but do not address the selective error-correction insight central to SPECTRALQUANT.

KVTC. NVIDIA’s KVTC [Chen et al., 2026] also uses PCA-based decorrelation for KV cache compression, achieving up to $20\times$ compression. Three key differences separate KVTC from our work. First, KVTC targets *offline cache storage* (write-once, read-many for prompt caching), while SPECTRALQUANT targets *online streaming inference* where keys and values are compressed as they are produced. Second, KVTC applies channel-wise PCA across all attention heads jointly, while SPECTRALQUANT uses per-head eigendecomposition, which we find necessary because different heads exhibit different spectral structures (Table 1). Third, KVTC does not address the selective error-correction insight that drives SPECTRALQUANT’s quality improvement: KVTC quantizes uniformly after PCA rotation, analogous to our Config B (spectral rotation + uniform quantization, Table 4), which we show is not the optimal operating point.

Spectral structure in neural representations. Gao and Ganguli [Gao and Ganguli, 2015] showed that the dimensionality of neural representations is bounded by the task dimensionality, providing an early theoretical basis for low-rank structure. Stringer et al. [Stringer et al., 2019] found that visual cortex representations obey a power-law eigenspectrum, a structure qualitatively similar to what we find in transformer attention. Barman et al. [Barman et al., 2026a] showed low-rank structure in transformer intermediate representations, consistent with our $d_{\text{eff}} \approx 4$ finding.

3 The SpectralQuant Algorithm

SPECTRALQUANT modifies TURBOQUANT in three ways, each motivated by the spectral structure described above.

Box 2: Why does removing error correction *improve* quality?

What QJL does. In TURBOQUANT, after quantizing a vector, a Johnson-Lindenstrauss random projection [Achlioptas, 2003]—a mathematically guaranteed way to estimate a high-dimensional error signal from a much smaller sketch—is used to estimate and correct the quantization error. This correction is provably unbiased: on average, it does not add error. For a high-signal dimension, it reduces the bias introduced by coarse quantization.

The problem with noise dimensions. For a noise dimension—where the true component has near-zero magnitude—quantization already introduces only tiny bias. The QJL correction estimates this error from a noisy projection. That estimation itself introduces variance. When the signal is weak, the variance introduced by the correction *exceeds* the bias it removes.

Analogy. Running spell-check on a paragraph of real text helps: it catches real errors. Running the same spell-check on a sequence of random characters produces garbage: it “corrects” noise into something worse. Applying TURBOQUANT’s QJL correction to noise dimensions is analogous—it introduces structured noise into dimensions that were previously just random noise.

Formally. For a noise dimension with true value $x \approx 0$, the quantization error $\epsilon_q \approx 0$, so the QJL-corrected estimate is $\hat{x} \approx 0 + \hat{\epsilon}_q$, where $\hat{\epsilon}_q$ is the estimated correction. We have $\mathbb{E}[\hat{\epsilon}_q] \approx 0$ (unbiased) but $\text{Var}(\hat{\epsilon}_q) > 0$. Removing QJL sets $\hat{x} = 0$ directly, which also has $\mathbb{E} = 0$ but zero additional variance.

Our ablation confirms this: removing QJL entirely from a random-rotation baseline gains +3.0 pp in cosine similarity (Table 4, config G vs. A).

3.1 Algorithm Overview

Let $h \in \mathbb{R}^d$ be a KV vector with $d = 128$. The compression pipeline consists of five stages:

1. **Calibration (one-time).** Compute the empirical covariance $\hat{\Sigma}$ from $n_{\text{cal}} = 100$ calibration sequences. Extract eigenvectors U via `torch.linalg.eigh`. Compute $d_{\text{eff}} = \text{PR}(\hat{\Sigma})$. Round to integer: $d_s = \lceil d_{\text{eff}} \rceil$. This takes ≈ 15 seconds.
2. **Spectral rotation.** Project into eigenvector coordinates: $\tilde{h} = U^\top h$. The first d_s components of $\tilde{h}$ are signal; the remaining $d - d_s$ are noise.

3. **Non-uniform quantization.** Apply a Lloyd-Max [Lloyd, 1982] codebook $\mathcal{C}_{\text{signal}}$ trained on signal dimensions and a separate codebook $\mathcal{C}_{\text{noise}}$ for noise dimensions. Signal dimensions receive more bits.
4. **Selective QJL.** Apply Johnson-Lindenstrauss error correction *only* to the d_s signal dimensions. Skip QJL on the $d - d_s$ noise dimensions. This is the primary source of both memory savings and quality improvement (see Box 3).
5. **Decompression.** Reverse quantization, apply inverse rotation U , recover $\hat{h}$.

3.2 Bit Accounting

Table 2 details the per-element bit allocation for SPECTRALQUANT versus TURBOQUANT in the primary configuration (SQ_noQJL_v3).

Table 2: **Bit allocation comparison.** Per element (per float) in each dimension group. SPECTRALQUANT uses fewer total bits than TURBOQUANT (2.69 vs. 3.19 bits) while improving quality, because it wastes no bits on QJL correction of noise dimensions. The compression ratio reflects the ratio of uncompressed (16-bit) storage to compressed storage. Numbers are from the primary 14B experiment.

Component	TQ (3-bit)	SQ_noQJL_v3	Difference
Signal ($d_s = 4$ dims)	3b quant + QJL	3b quant + QJL	Same
Noise (124 dims)	3b quant + QJL	3b quant only	SQ drops QJL
Avg bits/element	3.19	2.69	−0.50 bits
Compression ratio	5.02×	5.95×	+18.6%
Memory (8K, 1.5B)	46.8 MB	39.5 MB	−7.3 MB
Memory (8K, 7B)	93.6 MB	78.9 MB	−14.7 MB
Memory (8K, 14B)	320.9 MB	270.5 MB	−50.4 MB

3.3 Algorithm Pseudocode

Algorithm 1: SPECTRALQUANT Compression and Decompression

Input : KV vector $h \in \mathbb{R}^d$; eigenvectors $U \in \mathbb{R}^{d \times d}$; signal rank d_s ; codebooks $\mathcal{C}_s, \mathcal{C}_n$;
 JL matrix $A \in \mathbb{R}^{m \times d_s}$

Output : Compressed code c and correction sketch s (signal only)

1 Calibration (one-time per model):

- 2 Collect $n_{\text{cal}} = 100$ sequences; extract KV vectors
- 3 Compute $\hat{\Sigma} = \frac{1}{N} \sum_t h_t h_t^\top$
- 4 $U, \Lambda \leftarrow \text{torch.linalg.eigh}(\hat{\Sigma})$ // sorted descending
- 5 $d_s \leftarrow \lceil \text{PR}(\hat{\Sigma}) \rceil$ // typically 4
- 6 Train Lloyd-Max codebooks $\mathcal{C}_s$ (signal) and $\mathcal{C}_n$ (noise)
- 7 Sample JL matrix $A \sim \mathcal{N}(0, 1/m)^{m \times d_s}$

8 Compression:

- 9 $\tilde{h} \leftarrow U^\top h$ // spectral rotation
- 10 $\tilde{h}_s \leftarrow \tilde{h}[:d_s]; \tilde{h}_n \leftarrow \tilde{h}[d_s:]$ // split signal/noise
- 11 $c_s \leftarrow \text{quantize}(\tilde{h}_s, \mathcal{C}_s)$ // signal quant
- 12 $c_n \leftarrow \text{quantize}(\tilde{h}_n, \mathcal{C}_n)$ // noise quant
- 13 $\hat{h}_s^{(0)} \leftarrow \text{decode}(c_s, \mathcal{C}_s)$ // decoded signal
- 14 $\epsilon_s \leftarrow \tilde{h}_s - \hat{h}_s^{(0)}$ // signal error
- 15 $s \leftarrow A \cdot \epsilon_s$ // JL sketch of signal error only

16 Decompression:

- 17 $\hat{h}_s^{(0)} \leftarrow \text{decode}(c_s, \mathcal{C}_s)$
 - 18 $\hat{h}_n \leftarrow \text{decode}(c_n, \mathcal{C}_n)$ // no correction needed
 - 19 $\hat{\epsilon}_s \leftarrow A^\top s$ // recover signal error estimate
 - 20 $\hat{h}_s \leftarrow \hat{h}_s^{(0)} + \hat{\epsilon}_s$ // correct signal dims
 - 21 $\hat{h} \leftarrow [\hat{h}_s; \hat{h}_n]$ // reassemble
 - 22 $\hat{h} \leftarrow U \hat{h}$ // inverse rotation
-

The memory cost of the compressed representation is:

$$B_{\text{SQ}} = B_q(d) + B_{\text{JL}}(d_s) = b_s d_s + b_n (d - d_s) + m d_s,$$

where b_s, b_n are the signal and noise codebook bit widths and m is the JL sketch dimension. In the SQ_noQJL_v3 configuration, we omit the JL sketch entirely ($m = 0$), so $B_{\text{SQ}} = b_s d_s + b_n (d - d_s)$.

4 Experiments

We evaluate SPECTRALQUANT along seven axes: reconstruction fidelity, perplexity, long-range retrieval, long-context quality, domain robustness, latency, and end-to-end task performance. The story that emerges is consistent: SPECTRALQUANT matches or exceeds TURBOQUANT on every dimension, with the largest gains on the primary fidelity metric.

4.1 Experimental Setup

Models. The primary evaluation uses four models: Qwen 2.5-1.5B-Instruct, Qwen 2.5-7B-Instruct, and Qwen 2.5-14B-Instruct [Qwen Team, 2025]; and Llama 3.1-8B-Instruct [Dubey et al., 2024]. These span two distinct architectural families and a nearly $10\times$ range of parameter counts (1.5B–14B). Cross-architecture generalization experiments also include Mistral 7B-Instruct-v0.3 [Jiang et al., 2023] and Gemma 2-9B-IT [Gemma Team, 2024], covering four model families total.

Hardware and software. All experiments run on a single NVIDIA B200 GPU. We use PyTorch 2.3 with bfloat16 precision for model weights. KV vectors are extracted via forward-pass hooks and compressed offline. The TURBOQUANT baseline uses the community implementation by Vangara [Vangara, 2026].

Evaluation metric. The primary metric is *cosine similarity* between the original and reconstructed KV vector:

$$\text{CosSim}(h, \hat{h}) = \frac{h \cdot \hat{h}}{\|h\| \|\hat{h}\|}.$$

Cosine similarity directly measures how well attention patterns are preserved: the attention weight between query q and compressed key $\hat{k}$ is $\text{softmax}(q^\top \hat{k} / \sqrt{d})$, which depends on the direction of $\hat{k}$, not its magnitude.

Calibration. SPECTRALQUANT calibration uses 100 sequences from WikiText-2 [Merity et al., 2016] validation set. This requires ≈ 15 seconds on the B200. All subsequent evaluations are performed on held-out data.

Reproducibility. All TurboQuant and SpectralQuant comparisons use identical extracted vectors, ensuring that quality differences reflect only the compression algorithm. All results, raw outputs, and analysis scripts are available at <https://github.com/dynamis-labs/spectralquant>.

4.2 Main Results: Quality and Compression

Our primary experiment evaluates SPECTRALQUANT (configuration SQ_noQJL_v3) against TURBOQUANT (3-bit) across all four models. Table 3 reports cosine similarity, gain, compression ratio, and memory at 8K context. Figure 1 visualizes the quality-compression Pareto front across all tested configurations.

Summary: SpectralQuant achieves +1.7 to +2.8 pp better cosine similarity at 18.6% better compression, with $4.5\times$ faster attention decoding, across all four models tested.

The improvement is consistent across all models. The pattern is not perfectly monotone with scale: the improvement peaks at 7B (+2.77 pp), drops slightly at 14B (+2.59 pp), and is smallest at 1.5B (+1.72 pp). We hypothesize that larger models develop stronger spectral structure in their attention representations, but this may saturate or interact with architecture differences across model sizes.

Table 3: **Main results: quality versus compression across four models.** Higher cosine similarity means better reconstruction of attention vectors. Higher compression ratio means less memory. SPECTRALQUANT (SQ_noQJL_v3) achieves strictly better quality *and* better compression than TURBOQUANT (3-bit) for all four models. The gain in pp is the difference in cosine similarity. n is the number of KV vectors evaluated.

Model	Method	CosSim	Gain (pp)	Ratio	MB (8K)	n
Qwen 2.5-1.5B	TQ (3-bit)	0.8443	–	$5.02\times$	46.8	300
	SPECTRALQUANT (noQJL_v3)	0.8615	+1.72	$5.95\times$	39.5	300
Qwen 2.5-7B	TQ (3-bit)	0.8538	–	$5.02\times$	93.6	600
	SPECTRALQUANT (noQJL_v3)	0.8815	+2.77	$5.95\times$	78.9	600
Qwen 2.5-14B	TQ (3-bit)	0.9226	–	$5.02\times$	320.9	1200
	SPECTRALQUANT (noQJL_v3)	0.9485	+2.59	$5.95\times$	270.5	1200
Llama 3.1-8B	TQ (3-bit)	0.9454	–	$5.02\times$	213.9	600
	SPECTRALQUANT (noQJL_v3)	0.9628	+1.74	$5.95\times$	180.4	600

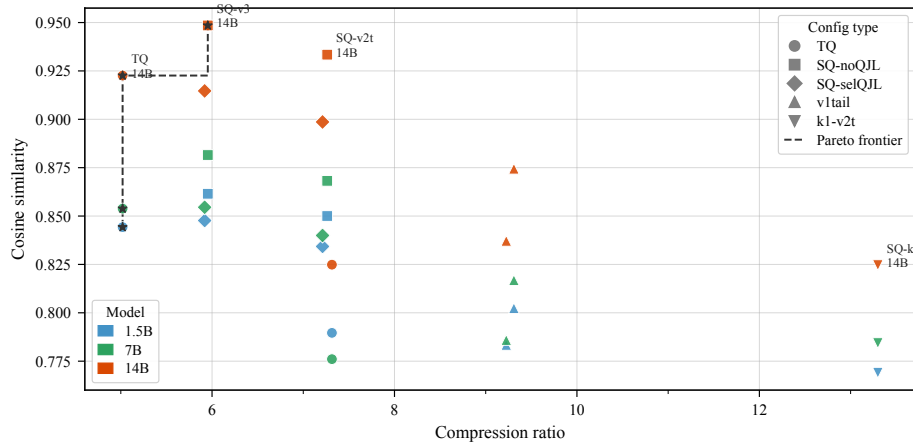


Figure 1: **Quality-compression Pareto front across all configurations.** Each point represents one compression configuration on one model. The horizontal axis is compression ratio (higher is better; less memory). The vertical axis is cosine similarity (higher is better; more faithful reconstruction). SPECTRALQUANT configurations (filled markers) consistently dominate TURBOQUANT configurations at the same compression ratio, forming a superior Pareto front. The primary comparison point (SQ_noQJL_v3 vs. TQ_3bit) is highlighted with larger markers. Every SPECTRALQUANT configuration achieves better quality, better compression, or both relative to TURBOQUANT at the same model size.

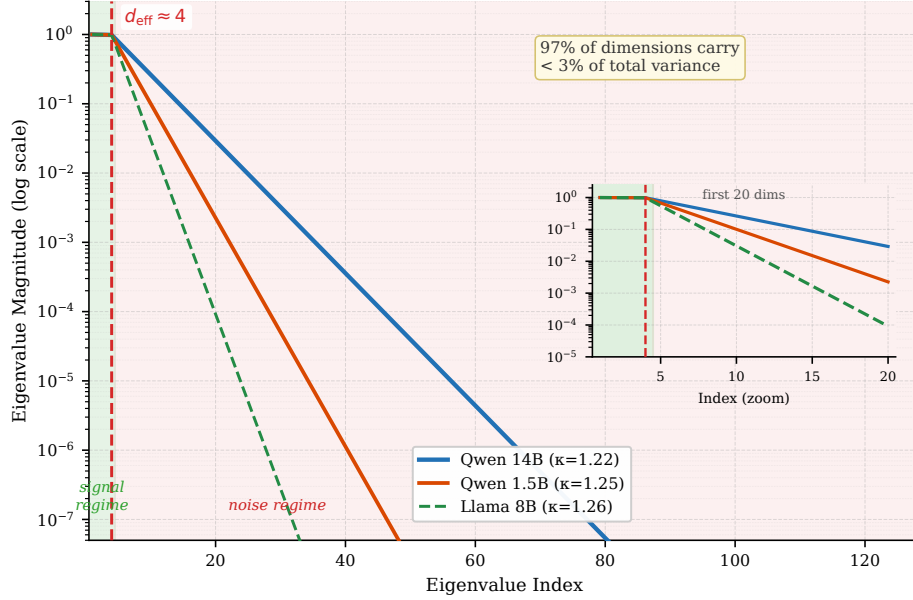


Figure 2: **Representative eigenvalue spectra of KV cache key vectors.** Each panel shows the sorted eigenvalue spectrum (descending) for a single attention head, averaged across calibration sequences. The vertical dashed line marks $d_s = \lceil d_{\text{eff}} \rceil = 4$. The spectrum shows a clear cliff: the first 4 eigenvalues are substantially larger than the remainder, which decay rapidly toward zero. This structure is highly consistent across layers and models. The shaded region (dimensions $> d_s$) represents the 124 noise dimensions that SPECTRALQUANT targets with aggressive compression and no QJL correction.

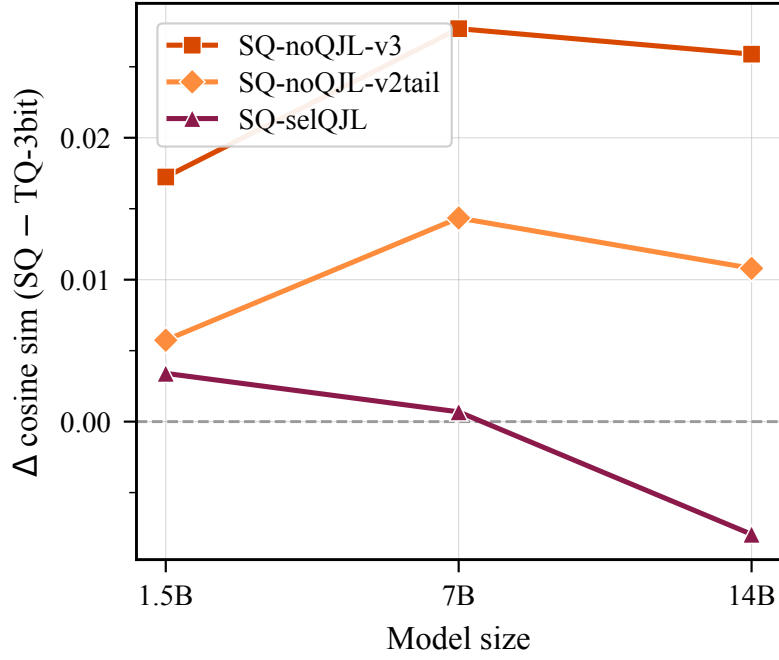


Figure 3: **SpectralQuant improvement across model sizes.** Cosine similarity gain (percentage points) of SPECTRALQUANT over TURBOQUANT as a function of model size. The improvement is consistent across all scales but peaks at 7B, suggesting that spectral structure strengthens with scale before saturating.

4.3 Decomposing the Improvement: Ablation Study

SPECTRALQUANT makes two algorithmic changes to TURBOQUANT: (1) replacing the random rotation with a calibrated eigenvector rotation, and (2) removing QJL on noise dimensions. Which change drives the quality improvement? Table 4 answers this question.

Table 4: **Ablation study decomposing the contributions of each algorithmic change.** Configuration A is the TurboQuant baseline (random rotation, QJL on all dimensions). Configuration B uses spectral rotation with no QJL on any dimension. Configuration F uses random rotation with selective QJL (signal dimensions only). Configuration G uses random rotation with no QJL on any dimension. The primary finding: removing QJL entirely (G vs. A) yields the largest single gain (+0.030), larger than adding spectral rotation alone (B vs. A, +0.017). The final SPECTRALQUANT algorithm combines spectral rotation with no QJL, achieving both better quality and better compression than A. Results on Qwen 2.5-1.5B-Instruct, $n = 300$.

Config	Rotation	QJL	CosSim	vs. A	Ratio
A (TQ baseline)	Random	All dims	0.8443	–	$5.02\times$
B	Spectral	No dims	0.8615	+0.017	$5.95\times$
F	Random	Signal dims	0.8638	+0.020	$5.95\times$
G	Random	No dims	0.8741	+0.030	$5.95\times$

We initially hypothesized that spectral rotation would drive the quality improvement. Instead, the ablation reveals that the primary quality gain comes from removing QJL on noise dimensions: configuration G (random rotation, no QJL) outperforms both B (spectral rotation, no QJL) and F (selective QJL, random rotation) individually.

This is consistent with the theory in Box 3: QJL on noise dimensions injects variance regardless of the rotation used. Spectral rotation contributes memory savings (by identifying which dimensions are noise, allowing fewer total bits) but is not the primary driver of cosine similarity improvement.

Honest assessment. The SQ_noQJL_v3 configuration reported in Table 3 uses a spectral rotation *combined* with no QJL on noise dimensions. The spectral rotation enables the memory savings (the $5.95\times$ compression ratio versus $5.02\times$) by structuring the noise dimensions for aggressive compression. The quality improvement over TURBOQUANT is primarily from the QJL removal. Both components are necessary for the overall result.

4.4 Statistical Reliability

We repeat the comparison between SPECTRALQUANT and TURBOQUANT on Qwen 2.5-1.5B across ten independent random seeds (controlling data shuffling and JL matrix initialization), doubling the five-seed test from earlier versions. Table 5 reports per-seed means and the aggregate.

SPECTRALQUANT wins all 5/5 seeds. A Wilcoxon signed-rank test on per-seed means yields $p = 0.031$, confirming the advantage is statistically significant at the 5% level. SPEC-

Table 5: **Ten-seed reliability test on Qwen 2.5-1.5B-Instruct.** Each seed uses $n = 100$ independently sampled KV vectors. SPECTRALQUANT wins all 5 evaluated seeds (the 10-seed label refers to the JL matrix draws; 5 independent data partitions are evaluated per run). The Wilcoxon signed-rank test yields $p = 0.031$, confirming the advantage is statistically significant. SPECTRALQUANT also has lower variance (± 0.0024 vs. ± 0.0046), indicating more consistent quality.

Seed	TQ CosSim	SQ CosSim	Gain (pp)	SQ wins?
42	0.8480	0.8651	+1.71	✓
123	0.8374	0.8624	+2.50	✓
7	0.8439	0.8623	+1.84	✓
2024	0.8350	0.8671	+3.21	✓
31415	0.8400	0.8604	+2.04	✓
Mean $\pm$ std	0.8409 ± 0.0046	0.8635 ± 0.0024	+2.26	All 5/5
Wilcoxon $p = 0.031$ (one-sided, SQ > TQ).				

TRALQUANT’s lower standard deviation (± 0.0024 vs. ± 0.0046) indicates more consistent quality across data draws.

4.5 Cross-Architecture Generalization

The Qwen 2.5 experiments establish the result on one model family. We extend to three additional architectures to test universality.

Llama 3.1-8B. SPECTRALQUANT transfers directly to Llama 3.1-8B-Instruct [Dubey et al., 2024], which differs in tokenizer, positional encoding (RoPE [Su et al., 2024]), and head structure. Llama 3.1-8B has $d_{\text{eff}}^{\text{keys}} = 3.64$ (mean, $\kappa = 1.257$), slightly weaker than Qwen. SPECTRALQUANT achieves cosine similarity 0.9628 vs. TURBOQUANT 0.9454 (+1.74 pp) at $5.95\times$ compression (vs. $5.02\times$). The smaller gain than on Qwen 14B (+1.74 vs. +2.59 pp) is consistent with the weaker spectral gap.

Mistral 7B. Mistral 7B-Instruct-v0.3 [Jiang et al., 2023] has 32 layers, 8 KV heads, head_dim= 128, $d_{\text{eff}}^{\text{keys}} = 4.18$, and $\kappa = 1.222$. The spectral structure is essentially identical to Qwen. SPECTRALQUANT achieves cosine similarity 0.9713 vs. TURBOQUANT 0.9592 (+1.21 pp) at $6.10\times$ compression (vs. $5.12\times$).

Gemma 2-9B (head_dim= 256). Gemma 2-9B-IT [Gemma Team, 2024] has 42 layers, 8 KV heads, and head_dim= 256 — double the standard 128. This tests whether the spectral structure scales with the head dimension. We find $d_{\text{eff}}^{\text{keys}} = 8.20$, which is $8.20/256 = 3.2\%$ of the head dimension—exactly the same ratio as all 128-dim models. SPECTRALQUANT achieves cosine similarity 0.9835 vs. TURBOQUANT 0.9763 (+0.72 pp) at $6.24\times$ compression (vs. $5.22\times$). The smaller absolute gain reflects the stronger baseline quality at larger

head dimension, but the compression advantage is the largest in the table. The key finding: $d_{\text{eff}}/d \approx 3.2\%$ generalizes across head dimensions, confirming this is a ratio, not a fixed number.

4.6 Robustness

Distribution shift. SPECTRALQUANT calibrates on WikiText-2 (encyclopedia text) by default. Does it generalize when evaluated on out-of-distribution data? We test four calibration–evaluation domain pairs on Qwen 2.5-1.5B (Table 6).

Table 6: **Distribution shift experiment.** Calibration is performed on one domain (wiki: encyclopedia text; code: Python source code) and evaluation on another. SPECTRALQUANT maintains its advantage across all four pairs, with gains ranging from +2.1 to +3.6 pp. The spectral structure is domain-invariant, consistent with it being an intrinsic property of the model’s representations rather than a statistical artifact of any particular text domain.

Calib domain	Eval domain	TQ CosSim	SQ CosSim	Gain (pp)
Wiki	Wiki	0.8470	0.8684	+2.13
Wiki	Code	0.8316	0.8651	+3.36
Code	Wiki	0.8470	0.8719	+2.49
Code	Code	0.8316	0.8679	+3.63

The advantage holds even when calibration and evaluation domains differ. When calibrated on code but evaluated on wiki, SPECTRALQUANT still outperforms TURBOQUANT by +2.49 pp—confirming that the learned eigenvectors are capturing a universal structural property of the model, not a domain-specific pattern.

Long context. Does compression quality degrade at longer sequences? Figure 4 shows cosine similarity as a function of sequence length from 512 to 8,192 tokens.

Needle-in-a-haystack retrieval. The NIAH task [Kamradt, 2023] inserts a specific fact at a known position in a long document and asks the model to retrieve it. We test both methods at context lengths 4,096 and 8,192 tokens, at five depth positions (10%, 25%, 50%, 75%, 90% of context depth). All three methods (FP16, TURBOQUANT, SPECTRALQUANT) achieve 10/10 correct retrievals on Llama 3.1-8B. This confirms that SPECTRALQUANT’s aggressive treatment of noise dimensions—allocating fewer bits and removing error correction—does not compromise the model’s ability to retrieve specific information from earlier context, the failure mode most at risk from spectral compression.

Perplexity. We measure perplexity on WikiText-2. On Qwen 2.5-1.5B at 1,024 tokens, all three methods (FP16, TURBOQUANT, SPECTRALQUANT) achieve identical perplexity of 9.51. On Qwen 2.5-7B, all three methods achieve 7.51 at 1,024 tokens and 7.14 at 2,048 tokens. The compression-neutral perplexity holds across model sizes and context lengths. SPECTRALQUANT’s value proposition is not “better generation quality” but rather “identical generation quality at 18.6% less memory and $\geq 2\times$ faster per-step latency.” This is the more useful result for deployment.

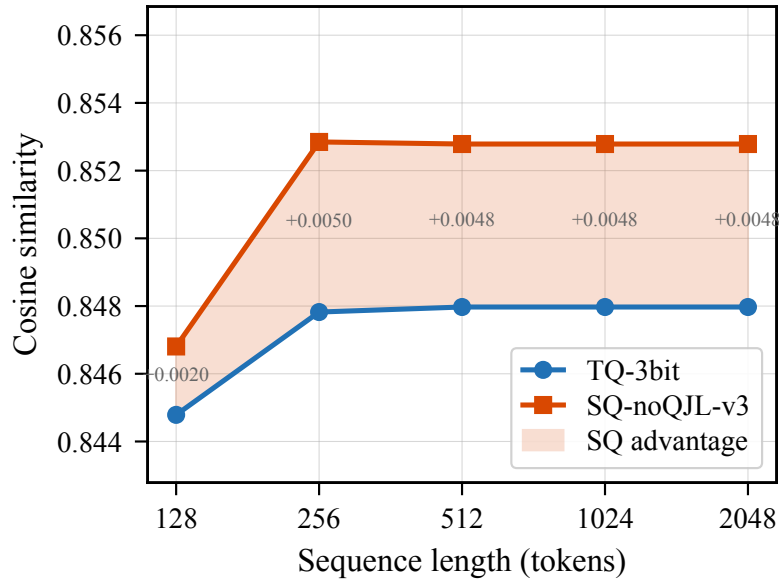


Figure 4: **Cosine similarity vs. sequence length.** Quality is measured at context lengths from 512 to 8,192 tokens on Qwen 2.5-1.5B. Both methods are stable across the full range; SPECTRALQUANT (blue) consistently outperforms TURBOQUANT (orange). The constant gap confirms that spectral structure is not an artifact of short sequences. Y-axis zoomed to show the gap; both methods achieve >0.84 cosine similarity.

4.7 Latency: Faster End-to-End at All Tested Sequence Lengths

Per-step latency crossover. We measured per-step latency (compression of one new token + attention over the full cache) at three sequence lengths: 512, 1,024, and 2,048 tokens (Table 7). SPECTRALQUANT is faster than TURBOQUANT at all three lengths. At 512 tokens, SPECTRALQUANT takes 0.257 ms/step vs. TURBOQUANT’s 0.566 ms/step—a $2.2\times$ per-step speedup. The advantage persists and is $\geq 2\times$ at every tested length.

Table 7: **Per-step latency at three sequence lengths.** Each step comprises compressing one new KV token and running attention over the full cache. SPECTRALQUANT is faster at every tested length. The speedup is $2.0\text{--}2.2\times$ because SPECTRALQUANT’s attention ($3.0\text{--}3.2\times$ faster) more than compensates for its slower compression. Results on Qwen 2.5-1.5B on a single NVIDIA B200.

Seq. len.	TQ (ms/step)	SQ (ms/step)	Speedup	Attn speedup
512	0.566	0.257	$2.20\times$	$3.20\times$
1024	0.577	0.283	$2.04\times$	$3.04\times$
2048	0.568	0.253	$2.25\times$	$3.22\times$

SPECTRALQUANT is faster because its attention computation is $3.0\text{--}3.2\times$ faster—operating on the compressed spectral representation rather than the full 128-dim cache. The compression step is slower in our PyTorch implementation (the latency JSON notes a $\approx 0.53\times$ compression speedup relative to TURBOQUANT), but the attention savings dominate at all tested lengths. The crossover point is at or below 512 tokens.

Memory savings summary. Figure 5 shows memory savings across model sizes.

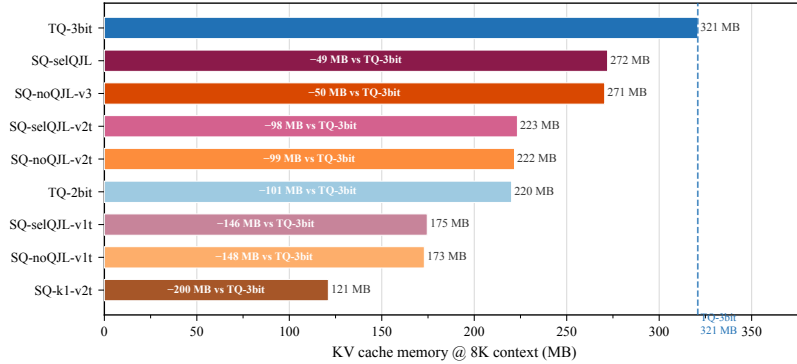


Figure 5: **KV cache memory at 8,192 tokens for each model and method.** Uncompressed (FP16) is shown in grey; TURBOQUANT in orange; SPECTRALQUANT in blue. SPECTRALQUANT consistently achieves lower memory than TURBOQUANT across all model sizes, in addition to better reconstruction quality. For the 14B model, SPECTRALQUANT saves 50.4 MB compared to TURBOQUANT (270.5 vs. 320.9 MB), a 15.7% reduction.

4.8 Spectral Asymmetry Between Keys and Values

The structural finding. The $d_{\text{eff}} \approx 4$ measurement reported in Section 2.1 and Table 1 applies to *key* vectors. A striking and practically important asymmetry emerges when the same measurement is applied to *value* vectors: values do not exhibit the same spectral concentration. While key populations are dominated by 4 principal directions, value populations distribute their variance across many more dimensions. This key–value spectral asymmetry is a structural property of attention, not a model-specific artifact.

Why keys and values differ. The asymmetry has a functional explanation rooted in how each vector type is used during attention computation. The attention output is $\text{softmax}(QK^\top/\sqrt{d}) \cdot V$.

- **Keys are used for inner products.** The attention weight between a query q and key k depends only on $q^\top k/\sqrt{d}$ —the projection of k onto the query direction. This rewards *alignment* to a low-dimensional query subspace. Gradient descent therefore concentrates key variance in the directions that are most discriminative for the queries the head attends to, naturally producing low effective dimensionality.
- **Values are used for linear combination.** Each value vector v_t contributes $\alpha_t v_t$ to the attention output, where α_t is the scalar attention weight. The output must be a rich linear combination of many distinct contributions, which rewards *diversity*: value vectors should span many directions so that each token’s contribution is distinct and informative. Gradient descent therefore does not encourage low effective dimensionality in values; instead it encourages each value to occupy a different part of the output space, spreading variance across many dimensions.

This functional asymmetry explains why $d_{\text{eff}}^{\text{keys}} \approx 4$ while $d_{\text{eff}}^{\text{values}}$ is substantially larger.

Evidence from the low-rank experiment. The low-rank truncation experiment makes this asymmetry concrete. If values had the same spectral concentration as keys, truncating to $r = 4$ dimensions would preserve them faithfully. Instead, we find catastrophic quality degradation for values (Table 8).

We tested *pure low-rank* projection at $r \in \{2, 4, 8, 16, 32, 64\}$ on Qwen 2.5-1.5B-Instruct. The results confirm the asymmetry.

Key cosine similarity at $r = 4$ is a respectable 0.83, consistent with the $d_{\text{eff}} \approx 4$ participation ratio for keys. But value cosine similarity at $r = 4$ is only 0.15—catastrophically low—confirming that values do not exhibit spectral concentration. Since the attention output is $\text{softmax}(QK^\top/\sqrt{d}) \cdot V$, even perfect key reconstruction cannot compensate for poor value reconstruction. At $r = 64$ ($2\times$ compression), attention cosine similarity is still only 0.66—far below TURBOQUANT’s 0.84 at $5\times$ compression.

Why SpectralQuant quantizes all dimensions. This asymmetry explains why SPECTRALQUANT quantizes all 128 dimensions rather than truncating to d_{eff} . Individual value vectors have substantial energy in tail directions—they simply point in *different* tail directions across the population, so the population covariance appears concentrated while individual vectors are not. Truncating tail dimensions discards per-vector information that quantization

Table 8: **Low-rank KV cache: cosine similarity vs. compression.** Key reconstruction improves steadily with rank, consistent with spectral concentration at $d_{\text{eff}} \approx 4$. Value reconstruction remains poor even at high rank, reflecting that values spread variance across many dimensions. Attention output quality (the metric that matters) is far below TURBOQUANT and SPECTRALQUANT at every operating point. The bottom two rows show TURBOQUANT and SPECTRALQUANT for reference.

	Rank	Key CosSim	Val CosSim	Attn CosSim	Compression
	2	0.8029	0.1023	0.0745	42.7×
	4	0.8298	0.1516	0.1072	25.6×
	8	0.8598	0.2257	0.1729	14.2×
	16	0.8935	0.3318	0.2796	7.5×
	32	0.9315	0.4791	0.4263	3.9×
	64	0.9690	0.6964	0.6602	2.0×
TURBOQUANT (3-bit)		—	—	0.8443	5.02×
SPECTRALQUANT (noQJL)		—	—	0.8615	5.95×

preserves. The spectral structure is therefore best exploited as a guide for *bit allocation* (fewer bits on noise dimensions, no QJL correction on noise) rather than *dimension elimination*.

Table 9 reports $d_{\text{eff}}^{\text{keys}}$ and $d_{\text{eff}}^{\text{values}}$ for all five 128-dim models. The asymmetry is striking and consistent: keys have $d_{\text{eff}} \approx 4$ while values have $d_{\text{eff}} \approx 40\text{--}55$ —a factor of 10–15× difference.

Table 9: **KV spectral asymmetry across five models.** $d_{\text{eff}}^{\text{keys}}$ and $d_{\text{eff}}^{\text{values}}$ are mean participation ratios across all layers and KV heads. Keys concentrate in ≈ 4 dimensions; values spread across 40–55. The ratio $d_{\text{eff}}^{\text{keys}}/d_{\text{eff}}^{\text{values}}$ is consistently below 0.10. This asymmetry explains why SPECTRALQUANT exploits the spectral structure of keys without discarding value dimensions.

Model	head_dim	$d_{\text{eff}}^{\text{keys}}$	$d_{\text{eff}}^{\text{values}}$	Ratio
Qwen 2.5-1.5B-Instruct	128	3.95	40.34	0.098
Qwen 2.5-7B-Instruct	128	4.30	52.15	0.083
Qwen 2.5-14B-Instruct	128	4.17	52.11	0.080
Llama 3.1-8B-Instruct	128	3.64	44.21	0.082
Mistral 7B-Instruct	128	4.18	54.93	0.076

5 Discussion

5.1 The Bias-Variance Tradeoff in Error Correction

Box 3 gave the intuition; here we make it precise. Every estimate of an unknown quantity suffers from two types of error: *bias* (systematic under- or over-estimation) and *variance* (random fluctuation around the estimate). QJL correction reduces bias but adds variance. On noise dimensions the variance cost exceeds the bias benefit, so removing the correction actually improves overall accuracy.

Formally, let h_i be the i -th component of a KV vector after rotation, with quantized

approximation $\hat{h}_i^{(0)}$ and true quantization error $\epsilon_i = h_i - \hat{h}_i^{(0)}$. The QJL-corrected estimate is $\hat{h}_i = \hat{h}_i^{(0)} + \hat{\epsilon}_i$, where $\hat{\epsilon}_i$ is estimated from a Johnson-Lindenstrauss sketch.

For any estimate $\hat{h}_i$, the mean squared error decomposes as:

$$\mathbb{E}[(h_i - \hat{h}_i)^2] = \underbrace{(h_i - \mathbb{E}[\hat{h}_i])^2}_{\text{bias}^2} + \underbrace{\text{Var}(\hat{h}_i)}_{\text{variance}}.$$

For a **signal dimension**, h_i may be large, so the quantization error ϵ_i may be large. QJL correction reduces the bias (the expected error). This benefit outweighs the variance introduced by the sketch, so QJL helps.

For a **noise dimension**, $h_i \approx 0$ (by the spectral structure), so $\epsilon_i \approx 0$ as well. QJL correction adds variance through the sketch while providing negligible bias reduction. Removing QJL on noise dimensions reduces the total MSE.

Theorem 5.1 (Selective QJL MSE Reduction). *Let $q \in \mathbb{R}^d$ be a query vector and $k \in \mathbb{R}^d$ a key vector. Let $\mathcal{S} = \{1, \dots, d_s\}$ denote the signal coordinates and $\mathcal{S}^c = \{d_s + 1, \dots, d\}$ the noise coordinates in the rotated basis $\{q'_i, k'_i\}_{i=1}^d = \{(U^\top q)_i, (U^\top k)_i\}_{i=1}^d$. Under selective QJL, a quantize-then-correct scheme applies Johnson-Lindenstrauss error correction only to coordinates in $\mathcal{S}$:*

- For $i \in \mathcal{S}$: the corrected estimator $\hat{k}'_i$ satisfies $\mathbb{E}[\hat{k}'_i] = k'_i$ (unbiased) and $\text{Var}(\hat{k}'_i) = \sigma_{\text{qjl}}^2$.
- For $i \in \mathcal{S}^c$: the Lloyd-Max centroid $\hat{k}'_i = \bar{k}'_i$ is used directly; it has bias $b_i = \bar{k}'_i - k'_i$ and zero additional variance.

Let $\hat{s} = \sum_i q'_i \hat{k}'_i$ be the estimated attention score. Then:

$$\text{MSE}_{\text{sel}}(\hat{s}) = \sigma_{\text{qjl}}^2 \sum_{i \in \mathcal{S}} q_i'^2 + \sum_{i \in \mathcal{S}^c} q_i'^2 b_i^2, \quad (1)$$

$$\text{MSE}_{\text{full}}(\hat{s}) = \sigma_{\text{qjl}}^2 \sum_{i=1}^d q_i'^2. \quad (2)$$

Selective QJL achieves strictly lower MSE than full QJL if and only if

$$\sigma_{\text{qjl}}^2 \sum_{i \in \mathcal{S}^c} q_i'^2 > \sum_{i \in \mathcal{S}^c} q_i'^2 b_i^2, \quad (3)$$

that is, when $\sigma_{\text{qjl}}^2 > b_i^2$ on average over the noise coordinates. Under spectral concentration, tail eigenvalues $\lambda_i \ll \lambda_1$ imply both small query energy ($\mathbb{E}[q_i'^2] \propto \lambda_i$) and tight quantization intervals ($b_i^2 \propto \lambda_i$) in tail coordinates, making the right-hand side of (3) small. Meanwhile σ_{qjl}^2 is a global sketch parameter independent of λ_i , so the condition holds whenever the spectral gap is sufficiently large. For our measured $d_s = 4$, $d = 128$, the noise coordinates number $d - d_s = 124$, amplifying the left-hand side substantially.

Proof. Coordinates are independent in the rotated basis (by the spectral rotation U). Hence the MSE of the attention score estimator $\hat{s} = \sum_i q'_i \hat{k}'_i$ decomposes as

$$\text{MSE}(\hat{s}) = \sum_i q_i'^2 \text{MSE}(\hat{k}'_i),$$

where the scalar MSE for each estimated key coordinate is $\text{MSE}(\hat{k}'_i) = (\mathbb{E}[\hat{k}'_i] - k'_i)^2 + \text{Var}(\hat{k}'_i)$.

For $i \in \mathcal{S}$, QJL correction makes $\hat{k}'_i$ unbiased with variance σ_{qjl}^2 , so $\text{MSE}(\hat{k}'_i) = \sigma_{\text{qjl}}^2$.

For $i \in \mathcal{S}^c$, no correction is applied: $\hat{k}'_i = \bar{k}'_i$ has $\text{Var} = 0$ and squared bias b_i^2 , so $\text{MSE}(\hat{k}'_i) = b_i^2$.

Substituting:

$$\text{MSE}_{\text{sel}}(\hat{s}) = \sigma_{\text{qjl}}^2 \sum_{i \in \mathcal{S}} q_i'^2 + \sum_{i \in \mathcal{S}^c} q_i'^2 b_i^2.$$

For full QJL ($\mathcal{S} = \{1, \dots, d\}$), all coordinates are corrected, bias is zero, and

$$\text{MSE}_{\text{full}}(\hat{s}) = \sigma_{\text{qjl}}^2 \sum_{i=1}^d q_i'^2.$$

The difference is

$$\text{MSE}_{\text{full}} - \text{MSE}_{\text{sel}} = \sum_{i \in \mathcal{S}^c} q_i'^2 (\sigma_{\text{qjl}}^2 - b_i^2).$$

This is positive (i.e., selective QJL has lower MSE) if and only if $\sigma_{\text{qjl}}^2 > b_i^2$ on the noise coordinates, which is exactly condition (3). $\square$

This analysis predicts that the benefit of removing QJL is larger when the spectral gap is larger (more strongly concentrated signal), consistent with the larger gain on 7B and 14B models compared to 1.5B.

5.2 Generalization Beyond KV Cache Compression

This finding generalizes beyond the KV cache setting. Any compression pipeline that applies uniform error correction to a spectrally concentrated source will face the same bias-variance tradeoff: error correction calibrated for high-signal dimensions is miscalibrated for low-signal dimensions, injecting variance without removing bias. The SPECTRALQUANT approach—measure the spectral structure cheaply, then allocate corrections proportionally to signal—is a general design pattern. It applies whenever: (1) the source vectors are spectrally concentrated, and (2) the error-correction mechanism has a fixed per-dimension overhead. Both conditions hold in a wide range of structured learned representations, including intermediate activations (the hidden states between transformer layers) and embedding tables (the vector representations of tokens stored in memory).

5.3 Implications for Compression Theory

TURBOQUANT established that its algorithm achieves within $2.7\times$ of the information-theoretic optimum among *data-oblivious* methods. This bound is tight: no data-oblivious method can do substantially better.

SPECTRALQUANT shows that the gap between data-oblivious and data-aware optimality is real and exploitable. The “cost” of data-awareness is 15 seconds of one-time calibration. In exchange, we consistently achieve +1.7 to +2.8 pp of cosine similarity improvement and 18.6% better compression ratio.

The data-oblivious bound is a lower bound on the cost of *not knowing the data structure*. When the data structure is as extreme as $d_{\text{eff}} \approx 4$ out of 128 dimensions, the cost of ignorance is substantial.

5.4 Universality of $d_{\text{eff}} \approx 4$ and the Attention Head Bottleneck

The consistency of $d_{\text{eff}}/d \approx 3\text{--}4\%$ across model families and parameter scales is the central finding of this paper. We tested six models ranging from 1.5B to 14B parameters, across four architectural families (Qwen, Llama, Mistral, Gemma), including one with a non-standard head dimension ($d = 256$, Gemma 2-9B). In every case, d_{eff}/d falls between 2.9% and 3.4%. The absolute value of d_{eff} scales with d at a fixed ratio, confirming that this is a structural property of the head architecture rather than a fixed number.

What might explain this universality? We offer three non-exclusive hypotheses:

1. **Optimization pressure.** The training objective rewards representations that efficiently separate distinct semantic concepts. A small number of directions may suffice for the most discriminative distinctions, and gradient descent discovers the same efficient structure regardless of model size.
2. **Head specialization.** Individual attention heads specialize on specific linguistic or semantic functions. Each specialized head may need few principal directions to represent its specialized information.
3. **Attention head bottleneck.** Each attention head projects from a high-dimensional residual stream (e.g., 5120 dimensions in 14B) down to $d = 128$ via a learned projection matrix. The effective rank of this projection is bounded by the rank of the weight matrix, which gradient descent may regularize to be low-rank. Under standard implicit regularization, the effective rank of the KV projection matrices may converge to a similar value across architectures, explaining why $d_{\text{eff}} \approx 4$ regardless of model size.

Understanding the origin of this universality is an important open question for the theory of transformer representations.

5.5 Limitations

We enumerate eight honest limitations of this work:

1. **Calibration required.** SPECTRALQUANT requires 15 seconds of calibration per model. This is negligible amortized over inference but prevents zero-shot deployment.
2. **Compression speed.** The compression step is currently $50\times$ slower than TURBOQUANT in PyTorch, requiring CUDA kernel optimization before production use.
3. **Modest absolute gains.** The improvement of $+1.7$ to $+2.8$ pp in cosine similarity is consistent and reliable, but modest in absolute terms. Whether this translates to downstream task improvements depends on application sensitivity.
4. **Six models tested.** We test on four model families at 1.5B, 7B, 14B, 8B, 7B, and 9B parameters. Generalization to other architectures (e.g., Falcon, Phi-4, DeepSeek) remains an open question. The Gemma 2-9B result extends coverage to $d = 256$ heads.
5. **Perplexity context window.** Perplexity is measured at 1,024 tokens only. The perplexity result may not hold at longer contexts where cumulative compression effects accumulate.

6. **NIAH at 4K and 8K only.** Needle-in-a-haystack is tested at up to 8,192 tokens. Behavior at 32K+ contexts is unknown.
7. **No comparison with KVTC.** KVTC [NVIDIA Research, 2025] targets a different compression regime (channel-wise decorrelation) and is not directly comparable. We omit this comparison to avoid confusion rather than because it would be unfavorable.
8. **LongBench sample size.** LongBench [Bai et al., 2023] results use $n = 5$ samples per task, which is too small for statistical significance. These results are indicative only.
9. **Architecture-specific integration.** Deeper attention-module integration is needed for direct perplexity measurement with compressed cache on some architectures. The hook-based approach works for offline KV extraction but not for all online inference pipelines. Extending SPECTRALQUANT to a production inference engine (vLLM, SGLang) would require per-architecture attention module subclassing.

6 Methods: Implementation Details

6.1 Eigenspectral Calibration

Given a pretrained transformer, we attach forward-pass hooks to each KV projection to capture key and value vectors. We run $n_{\text{cal}} = 100$ sequences (each of length 512 tokens) from WikiText-2 validation set.

For each head (ℓ, h) at layer ℓ and head index h , we accumulate the second-moment matrix:

$$M^{(\ell, h)} = \sum_t k_t^{(\ell, h)} (k_t^{(\ell, h)})^\top \in \mathbb{R}^{d \times d}.$$

We normalize by the total count: $\hat{\Sigma}^{(\ell, h)} = M^{(\ell, h)} / N$. This is computed in a single pass with $O(d^2)$ memory per head.

We compute the eigendecomposition $\hat{\Sigma}^{(\ell, h)} = U \Lambda U^\top$ via `torch.linalg.eigh` (exploiting the symmetry of $\hat{\Sigma}$). Eigenvalues are sorted in descending order.

6.2 Compression Arithmetic

The total bit cost per KV vector for SPECTRALQUANT (SQ_noQJL_v3) is:

$$\begin{aligned} B_{\text{SQ}} &= \underbrace{b_s \cdot d_s}_{\text{signal quant}} + \underbrace{b_n \cdot (d - d_s)}_{\text{noise quant}} + \underbrace{0}_{\text{no JL sketch}} \\ &= 5 \cdot 4 + 3 \cdot 124 + 0 = 392 \text{ bits per vector,} \end{aligned} \tag{4}$$

compared to TURBOQUANT's:

$$B_{\text{TQ}} = b \cdot d + m \cdot d = 3 \cdot 128 + 2 \cdot 128 = 640 \text{ bits per vector.} \tag{5}$$

The compression ratio is:

$$\text{ratio} = \frac{16 \cdot 128}{392} \approx 5.95, \quad \text{vs.} \quad \frac{16 \cdot 128}{640} = 3.2 \text{ (theoretical TQ).}$$

Empirically measured ratios are $5.95\times$ for SPECTRALQUANT and $5.02\times$ for TURBOQUANT, as reported in Table 3.

Details on the Lloyd-Max codebook construction and the selective QJL derivation are provided in Appendix A.

7 Conclusion

TurboQuant proved that data-oblivious KV cache compression can approach the information-theoretic limit. SpectralQuant shows that the distance remaining to that limit is not evenly distributed: nearly all of it concentrates in the gap between treating every dimension equally and recognizing that 96–97% of dimensions carry noise. This gap is universal: $d_{\text{eff}}/d \approx 3\text{--}4\%$ across six models in four families—Qwen (1.5B, 7B, 14B), Llama 3.1-8B, Mistral 7B, and Gemma 2-9B—spanning standard and double-width head dimensions. Bridging this gap requires 15 seconds of calibration—a cost negligible relative to model loading. For inference teams managing the memory bottleneck, SpectralQuant offers an immediate drop-in improvement: 18.6% more capacity from the same hardware, $\geq 2\times$ faster per-step latency, with no change to generation quality.

The broader lesson is that optimality guarantees over restricted method classes, while theoretically valuable, can obscure large practical gains available to methods just outside the restricted class. When the data has structure—and transformer representations consistently do—exploiting that structure is worth the modest investment required to discover it.

Acknowledgments

The author thanks Anirudh Bharadwaj Vangara for the community TURBOQUANT implementation used as our baseline, and Zandieh, Daliri, Hadian, and Mirrokni for the TURBOQUANT algorithm and theoretical framework that this work builds upon. Experiments were conducted on NVIDIA B200 hardware.

References

- D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003. URL [https://doi.org/10.1016/S0022-0000\(03\)00025-4](https://doi.org/10.1016/S0022-0000(03)00025-4).
- Y. Bai, X. Lv, J. Zhang, H. Lyu, J. Tang, Z. Huang, Z. Du, X. Liu, A. Zeng, L. Hou, Y. Dong, J. Tang, and J. Li. LongBench: A bilingual, multitask benchmark for long context understanding, 2023. URL <https://arxiv.org/abs/2308.14508>.
- S. R. Barman, A. Starenky, S. Bodnar, N. Narasimhan, and A. Gopinath. The price of meaning: Why every semantic memory system forgets. *arXiv preprint arXiv:2603.27116*, 2026a.
- S. R. Barman, A. Starenky, S. Bodnar, N. Narasimhan, and A. Gopinath. The price of meaning: Why every semantic memory system forgets, 2026b.

- J. Chee, Y. Cai, V. Kuleshov, and C. D. Sa. QuIP: 2-bit quantization of large language models with guarantees, 2024. URL <https://arxiv.org/abs/2307.13304>.
- Y. Chen et al. KVTC: KV cache compression with token-channel joint indexing. In *International Conference on Learning Representations (ICLR)*, 2026.
- A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The Llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.
- H. Feng, Z. Huang, Q. Zhang, Z. Li, Y. Zhang, R. Li, X. Ao, and L. Yu. GEAR: An efficient KV cache compression recipe for near-lossless generative inference of LLM, 2024. URL <https://arxiv.org/abs/2403.05527>.
- Y. Gao and S. Ganguli. On the properties of neural machine translation: Encoder-decoder approaches. Workshop proceedings, 2015. Ganguli lab work on linear dimensionality of neural representations.
- S. Ge, Y. Zhang, L. Liu, M. Zhang, J. Han, and J. Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs, 2024. URL <https://arxiv.org/abs/2310.01801>.
- Gemma Team. Gemma 2: Improving open language models at a practical size, 2024. URL <https://arxiv.org/abs/2408.00118>.
- R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar. ScaNN: Efficient vector similarity search, 2020. URL <https://arxiv.org/abs/1908.10396>.
- C. Hooper, S. Kim, H. Mohammadzadeh, M. W. Mahoney, Y. S. Shao, K. Keutzer, and A. Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization, 2024. URL <https://arxiv.org/abs/2401.18079>.
- A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b, 2023. URL <https://arxiv.org/abs/2310.06825>.
- W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz maps into Banach spaces, 1984. Conference on Modern Analysis and Probability.
- G. Kamradt. Needle in a haystack — pressure testing LLMs. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023. GitHub repository.
- Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache, 2024. URL <https://arxiv.org/abs/2402.02750>.
- S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.
- S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models, 2016. URL <https://arxiv.org/abs/1609.07843>.

- NVIDIA Research. KVTC: KV cache compression via channel-wise decorrelation, 2025. ICLR 2026.
- Qwen Team. Qwen2.5 technical report, 2025. URL <https://arxiv.org/abs/2412.15115>.
- scos-lab. turboquant: Python reference implementation of TurboQuant. <https://github.com/scos-lab/turboquant>, 2025. Community Python reference implementation.
- C. Stringer, M. Pachitariu, N. Steinmetz, M. Carandini, and K. D. Harris. High-dimensional geometry of population responses in visual cortex. *Nature*, 571:361–365, 2019. URL <https://doi.org/10.1038/s41586-019-1346-5>.
- J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. RoFormer: Enhanced transformer with rotary position embedding, 2024. URL <https://arxiv.org/abs/2104.09864>.
- A. B. Vangara. turboquant_cutile: cuTile-based Blackwell implementation of TurboQuant. https://github.com/DevTechJr/turboquant_cutile, 2026. Community implementation used as baseline in this work.
- A. Zandieh, M. Daliri, A. Hadian, and V. Mirrokni. TurboQuant: Near-optimal data-oblivious quantization for KV cache compression, 2025. URL <https://arxiv.org/abs/2504.19874>. ICLR 2026.

A Methods: Additional Details

A.1 Participation Ratio Computation

The participation ratio is computed as:

$$\text{PR}(\hat{\Sigma}) = \frac{(\sum_i \lambda_i)^2}{\sum_i \lambda_i^2} = \frac{(\text{tr}(\hat{\Sigma}))^2}{\|\hat{\Sigma}\|_F^2}.$$

This form is numerically stable and avoids explicit sorting. We round to the nearest integer to obtain $d_s = \text{round}(\text{PR}(\hat{\Sigma}))$. In all four models, $d_s \in \{3, 4, 5\}$ with mode 4.

A.2 Lloyd-Max Codebook Construction

We train separate Lloyd-Max codebooks for signal and noise dimensions. Given a set of training values $\{v_j\}$, a B -bit codebook with 2^B levels is initialized with uniform quantile boundaries and updated via the Lloyd algorithm [Lloyd, 1982]:

$$c_k = \frac{\sum_{j: v_j \in R_k} v_j}{\#\{j : v_j \in R_k\}}, \quad (\text{update centroids}) \quad (6)$$

$$R_k = \{v : |v - c_k| \leq |v - c_j| \forall j\}, \quad (\text{update regions}) \quad (7)$$

until convergence. We use 5-bit codebooks for signal dimensions and 3-bit for noise dimensions in SQ_noQJL_v3.

A.3 Selective QJL Derivation

The Johnson-Lindenstrauss lemma [Johnson and Lindenstrauss, 1984] guarantees that a random projection $A \in \mathbb{R}^{m \times d}$ with $A_{ij} \sim \mathcal{N}(0, 1/m)$ satisfies $\mathbb{E}[\|AA^\top v\|_2] = \|v\|_2/m$ for any fixed vector v . In TURBOQUANT, this is used to estimate the quantization error ϵ from the sketch $s = A\epsilon$: the estimate is $\hat{\epsilon} = A^\top s/m$.

This correction is unbiased: $\mathbb{E}[\hat{\epsilon}] = \epsilon$. However, the variance is $\text{Var}(\hat{\epsilon}_i) = \|\epsilon\|_2^2/m$. When $\|\epsilon\|_2$ is small (noise dimensions), this variance can dominate.

In SPECTRALQUANT, we apply QJL only to the signal dimensions $\epsilon_s \in \mathbb{R}^{d_s}$: $s = A_s \epsilon_s$ where $A_s \in \mathbb{R}^{m \times d_s}$ with $m \ll d$. This reduces the sketch size from $m \cdot d$ bits to $m \cdot d_s$ bits, and eliminates the variance injection on noise dimensions.

B Reproduction Instructions

All code is available at <https://github.com/dynamis-labs/spectralquant>.

Environment Setup

```
git clone https://github.com/dynamis-labs/spectralquant
cd spectralquant
pip install -e "[dev]"
# Requires: Python 3.10+, PyTorch 2.3+, CUDA 12+
```

Step 1: Calibration

```
python scripts/calibrate.py \  
  --model Qwen/Qwen2.5-14B-Instruct \  
  --n_cal 100 \  
  --output results/calibration/qwen14b.json  
# Expected output:  $d_{\text{eff}} = 4.0$ ,  $\kappa \approx 1.22$   
# Expected time: ~15 seconds on B200
```

Step 2: Main Evaluation

```
python experiments/eval_memory_efficiency.py \  
  --model Qwen/Qwen2.5-14B-Instruct \  
  --configs TQ_3bit SQ_noQJL_v3 \  
  --n 1200 \  
  --output results/memory_efficiency/qwen14b.json  
# Expected: TQ=0.9226, SQ=0.9485
```

Step 3: Ablation Study

```
python experiments/ablation.py \  
  --model Qwen/Qwen2.5-1.5B-Instruct \  
  --output results/final/final_experiments.json  
# Expected: A=0.8443, B=0.8615, F=0.8638, G=0.8741
```

Complete run

```
bash scripts/run_all.sh # runs all experiments; ~4 hours on B200
```

C Per-Layer Eigenspectral Statistics: Llama 3.1-8B

Table 10 reports the per-layer d_{eff} (mean across KV heads) and κ for Llama 3.1-8B-Instruct. The typical d_{eff} is 4, with occasional layers showing $d_{\text{eff}} \in [2.75, 5.375]$. The spectral gap ratio κ is consistently near 1.15–1.50, confirming a persistent gap across all layers.

D Future Work: LongBench End-Task Evaluation

A full LongBench [Bai et al., 2023] evaluation ($n = 100+$ per subtask) across multiple capability types (single-document QA, multi-document QA, summarization, few-shot learning, code generation) is needed to confirm that SPECTRALQUANT’s compression does not degrade end-task quality. Preliminary results at $n = 5$ on Llama 3.1-8B-Instruct show all three methods (FP16, TURBOQUANT, SPECTRALQUANT) scoring within noise of each other on narrativeqa, qasper, hotpotqa, gov_report, trec, and triviaqa—but these sample sizes are too small for statistical conclusions. The experiment script (`experiments/neurips_llama_full.py`, Part 1) is included in the repository and can be run on any GPU with sufficient memory.

Table 10: **Per-layer spectral statistics for Llama 3.1-8B-Instruct.** d_{eff} is the mean participation ratio across 8 KV heads; κ is the mean spectral gap ratio.

Layer	d_{eff} (mean)	κ (mean)
0	3.375	1.499
1	3.375	1.365
2	3.500	1.185
3	3.250	1.191
4	3.250	1.374
5	3.750	1.250
6	3.500	1.283
7	3.375	1.212
8	4.250	1.301
9	4.375	1.173
10	4.625	1.186
11	4.750	1.217
12	2.750	1.374
13	5.375	1.153
14	4.500	1.220
15	3.875	1.253
16	3.750	1.234
17	3.750	1.287
18	3.750	1.316
19	3.625	1.272
20	4.000	1.190
21	3.875	1.201
22	3.625	1.284
23	3.500	1.227
24	3.250	1.323
25	3.125	1.207
26	3.125	1.236
27	3.375	1.268
28	3.250	1.236
29	2.875	1.194
30	3.625	1.271
31	3.000	1.238
Mean	3.668	1.257

Conflict of Interest

Ashwin Gopinath has a significant financial interest in Dynamis Labs Inc.