TurboQuant: Online Vector Quantization with Near-Optimal Distortion Rate

Amir Zandieh [1] Majid Daliri [1,2] Majid Hadian [1] Vahab Mirrokni [1]

[1]Google [2]New York University

Why vector quantization matters now

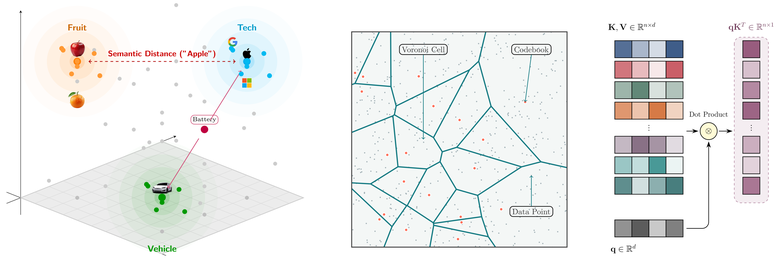


Figure 1. Traditional offline vector quantization methods assume quantization time is not a constraint, relying on the computationally expensive training of learned codebooks. However, in dynamic environments where latency is critical, online quantization is the ideal paradigm, as quantization time directly limits system throughput. TurboQuant addresses this bottleneck by providing a strictly online, training-free vector quantization approach that enables extreme compression without the overhead of offline learning.

Method overview

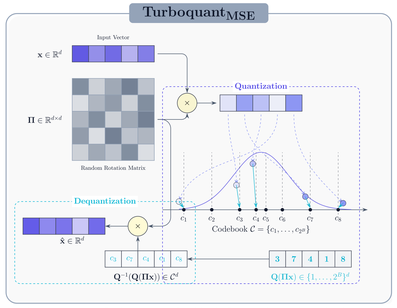


Figure 2. $\text{TurboQuant}_{\text{MSE}}$ quantizes vectors coordinate-wise using a fixed, precomputed codebook. Because the coordinate distribution is known exactly a priori, this codebook can be generated and stored in advance without data-dependent calibration.

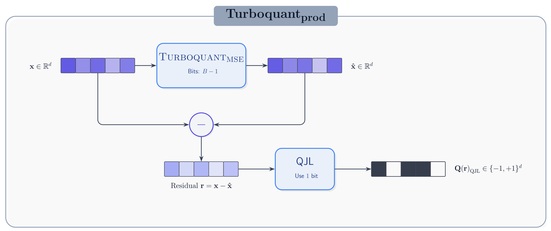


Figure 3. $\text{TurboQuant}_{\text{PROD}}$ combines rotation-based scalar quantization, $B-1$ bit $\text{TurboQuant}_{\text{MSE}}$, and a 1-bit residual correction to achieve near-optimal distortion and unbiased inner product estimation. As shown, $\text{TurboQuant}_{\text{MSE}}$ round-trips to produce the quantized vector $\tilde{\mathbf{x}}$, passing the residual $\mathbf{r} = \mathbf{x} - \tilde{\mathbf{x}}$ to QJL.

TurboQuant : KV cache Quantization

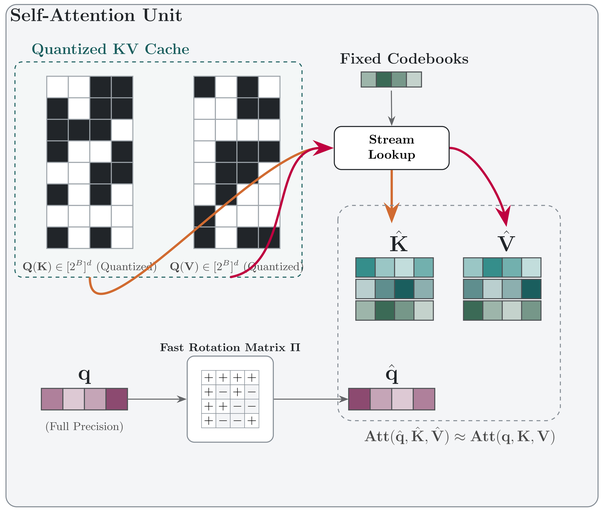


Figure 4. TurboQuant attention data flow. Quantized KV-cache is decoded into $\hat{\mathbf{K}}$ and $\hat{\mathbf{V}}$ via Stream Lookup, while the query $\mathbf{q}$ is rotated to $\hat{\mathbf{q}}$ using a random rotation matrix. This reduces memory bottlenecks, efficiently approximating $\text{Att}(\mathbf{q}, \mathbf{K}, \mathbf{V})$.

Near-optimal distortion guarantees

Theorem. For any unit vector $x \in \mathbb{S}^d$ and any bit-width b:

(1) Mean-squared error. The MSE stage of TurboQuant satisfies

$$D_{\text{mse}} \leq \sqrt{\frac{3\pi}{2}} 4^{-b}.$$

(2) Inner-product estimation. For any query vector $y \in \mathbb{R}^d$, the two-stage estimator is unbiased and satisfies

$$D_{\text{prod}} \leq \sqrt{\frac{3\pi}{2}} \frac{\|y\|_2^2}{d} 4^{-b}.$$

(3) Information-theoretic lower bound. Any randomized b-bit vector quantizer must incur worst-case distortion at least

$$D_{\text{mse}} \geq 4^{-b} \frac{d}{d-1}, \qquad D_{\text{prod}} \geq \frac{\|y\|_2^2}{d} 4^{-b} \frac{d}{d-1}.$$

Therefore, TurboQuant achieves the optimal distortion *rate* and is within a factor $\sqrt{\frac{3\pi}{2}} \approx 2.17$ of the lower bound.

Links

Paper: arxiv.org/abs/2504.19874

Related official sources: Google Research blog, QJL paper.

KV-cache results

TurboQuant pushes KV-cache compression hard while keeping downstream quality essentially intact.

- On **Needle-In-A-Haystack**, TurboQuant matches full precision at more than 4× compression.
- On **LongBench-E**, it is essentially quality-neutral at 3.5 bits/channel and degrades only slightly at 2.5 bits/channel.

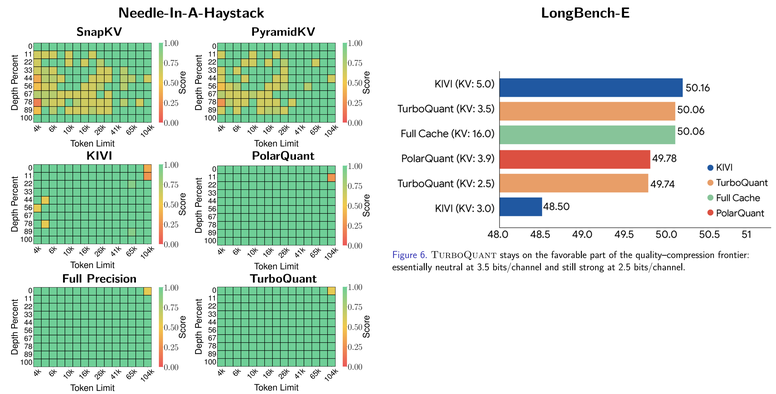
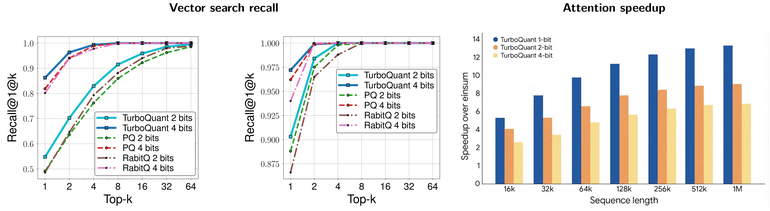


Figure 5. Needle-In-A-Haystack on `Llama-3.1-8B-Instruct`.

Figure 6. TurboQuant stays on the favorable part of the quality–compression frontier: essentially neutral at 3.5 bits/channel and still strong at 2.5 bits/channel.

Vector search & systems results

Takeaway: stronger recall than training-based baselines, plus real deployment speedups.



(a) Glove, $d = 200$ (b) OpenAI3, $d = 3072$

Figure 7. TurboQuant performs well without requiring dataset-specific training, and its systems implementation delivers strong attention-logit speedups.

References

[1] A. Zandieh, M. Daliri, and I. Han. *QJL: 1-Bit Quantized JL Transform for KV Cache Quantization with Zero Overhead.* AAAI 2025 / arXiv 2406.03482.

[2] Google Research Blog. *TurboQuant: Redefining AI efficiency with extreme compression.* March 24, 2026.