KYUNG HEE UNIVERSITY

Efficient Multimodal Intelligence Lab

TurboQuant: Online Vector Quantatization with Near-Optimal Distortion Rate

July 17, 2026

Dongwon Lee

Agenda

1. Summary
2. Preliminaries
3. Problem Definition
4. Proposed Methods
5. Experiments
6. Conclusion
7. Discussion
8. Future Work

Summary (1/1)

The Core of TURBOQUANT

- **MSE Optimal Vector Quantizer(PolarQuant)**
  - optimal distortion rate in terms of mean-squared error(MSE)
- **1-bit Residual Quantization(QJL)**
  - an unbiased and low-distortion inner product quantizer
- **Theoretical Guarantee**
  - MSE, Inner Product Objectives
  - Provably near-optimal within distortion lower bound

-

Summary (1/1)

Abstract

TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate

Amir Zandieh
Google Research
zandieh@google.com

Majid Daliri
New York University
daliri.majid@nyu.edu

Majid Hadian
Google DeepMind
majidh@google.com

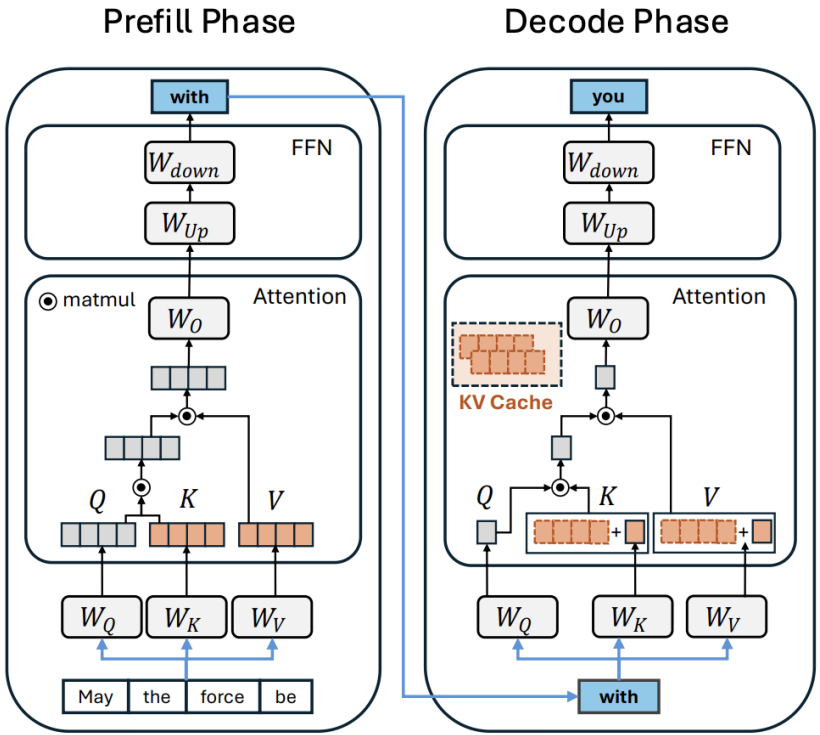
Vahab Mirrokni
Google Research
mirrokni@google.com

Abstract

Vector quantization, a problem rooted in Shannon's source coding theory, aims to quantize high-dimensional Euclidean vectors while minimizing distortion in their geometric structure. We propose TurboQuant to address both mean-squared error (MSE) and inner product distortion, overcoming limitations of existing methods that fail to achieve optimal distortion rates. Our data-oblivious algorithms, suitable for online applications, achieve near-optimal distortion rates (within a small constant factor) across all bit-widths and dimensions. TurboQuant achieves this by randomly rotating input vectors, inducing a concentrated Beta distribution on coordinates, and leveraging the near-independence property of distinct coordinates in high dimensions to simply apply optimal scalar quantizers per each coordinate. Recognizing that MSE-optimal quantizers introduce bias in inner product estimation, we propose a two-stage approach: applying an MSE quantizer followed by a 1-bit Quantized JL (QJL) transform on the residual, resulting in an unbiased inner product quantizer. We also provide a formal proof of

Preliminaries (1/4)

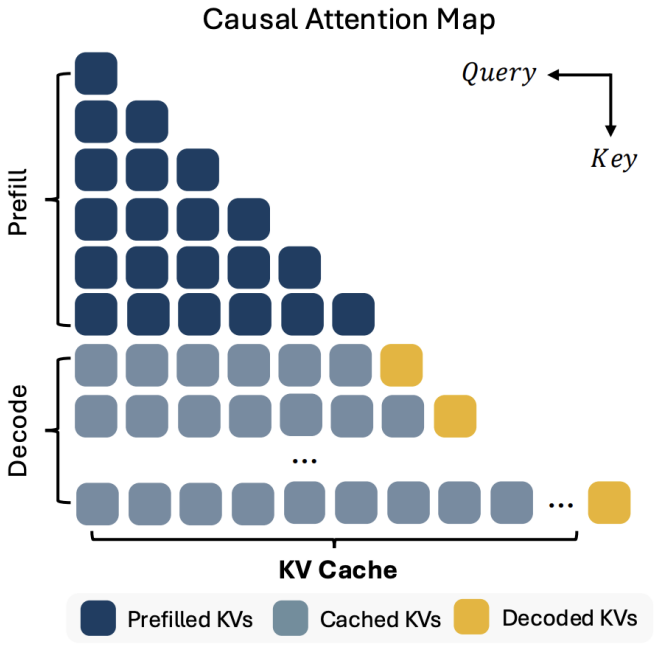
KV-Cache



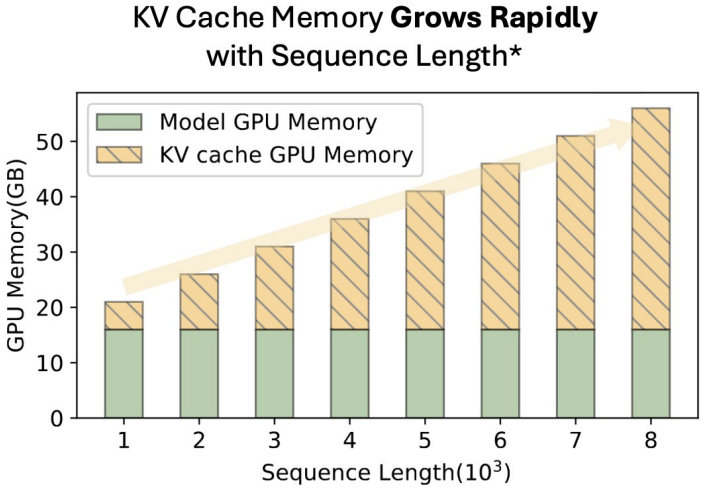
KV Cache prevents **re-computation** of hidden states during auto-regressive decoding.

Preliminaries (2/4)

KV-Cache Problem & Quantization



KV Cache size **grows linearly** with context length, leading to memory bottleneck



Model weights and KV cache memory consumption across context length* (LLaMA-3.1-8B, batch 10)

Preliminaries (3/4)

KV-Cache Problem & Quantization

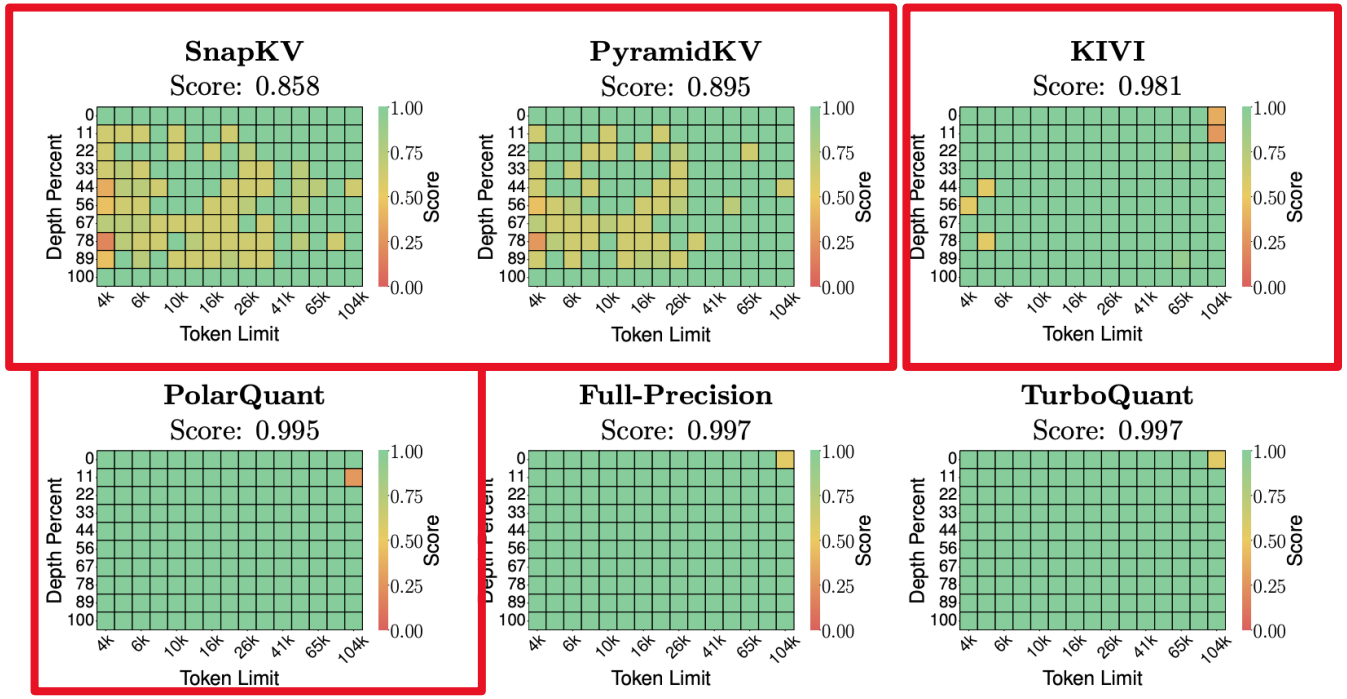


Figure 4: Evaluation of `Llama-3.1-8B-Instruct` on the "Needle-In-A-Haystack" test, where a model must retrieve a hidden sentence from long-context sequences. While some methods struggle with recall, TURBOQUANT, despite being more than 4× quantized, achieves the same exact performance as the uncompressed baseline.

Preliminaries (4/4)

Lossy vs Lossless Compression

- **TurboQuant is a lossy compression**
  - minimum distortion + reduce precision

Problem Definition

Goal

- **Goal:** Design a quantization map $Q : \mathbb{R}^d \to \{0,1\}^B$
  Inverse m: $Q^{-1} : \{0,1\}^B \to \mathbb{R}^d$

- **Objective:** minimize distortion

$$\textbf{(MSE)} \qquad D_{\texttt{mse}} := \mathop{\mathbb{E}}_{Q}\left[\left\|\boldsymbol{x} - Q^{-1}\left(Q(\boldsymbol{x})\right)\right\|_2^2\right]$$

$$\textbf{(inner-prod error)} \qquad D_{\texttt{prod}} := \mathop{\mathbb{E}}_{Q}\left[\left|\langle \boldsymbol{y}, \boldsymbol{x}\rangle - \langle \boldsymbol{y}, Q^{-1}\left(Q(\boldsymbol{x})\right)\rangle\right|^2\right].$$

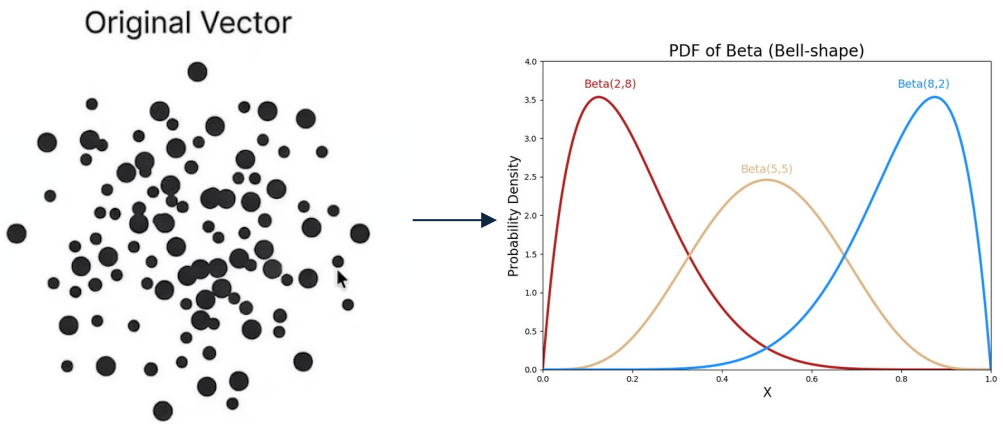
$$\textbf{(unbiased inner-prod)} \qquad \mathop{\mathbb{E}}_{Q}\left[\langle \boldsymbol{y}, Q^{-1}\left(Q(\boldsymbol{x})\right)\rangle\right] = \langle \boldsymbol{y}, \boldsymbol{x}\rangle.$$

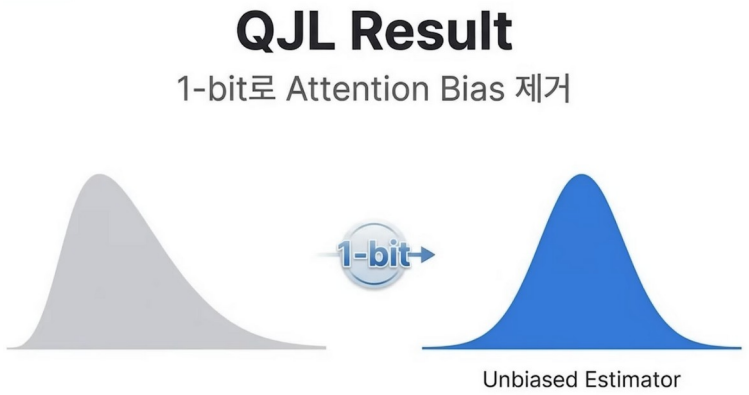
- **Computationally efficient quantizers $Q_{\texttt{mse}}$ and $Q_{\texttt{prod}}$ that achieve optimal bounds, for any given bit-width b.**

Proposed Methods (1/3)

Two-stage Process

- **MSE Optimal Vector Quantizer**
  - optimal distortion rate in terms of mean-squared error(MSE)
- **1-bit Residual Quantization(*QJL*)**
  - an unbiased and low-distortion inner product quantizer





Source: Wikipedia, Beta Distribution

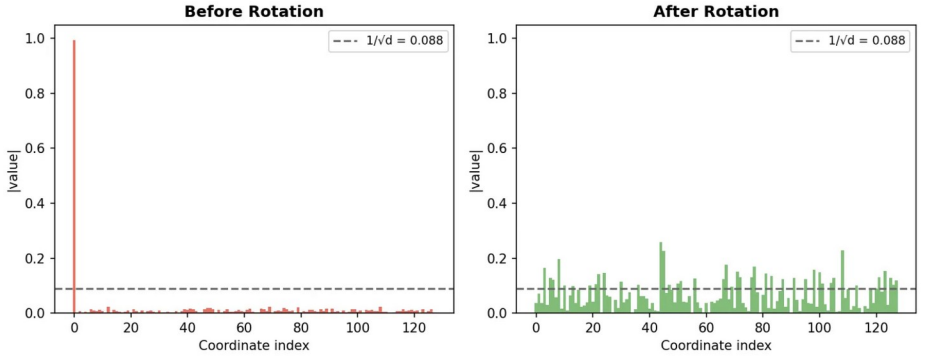
Proposed Methods (2/3)

MSE Optimized TurboQuant

- **Goal: Minimize MSE distortion** $D_{\texttt{mse}} := \mathop{\mathbb{E}}_{Q}\left[\left\|\boldsymbol{x} - Q^{-1}\left(Q(\boldsymbol{x})\right)\right\|_2^2\right]$

- **Step1: Random Rotation via QR**

- inducing a Beta distribution on each coordinate



- **Step 2: Max-Lloyd Scalar Quantization**
  - Optimal Scalar Quantizer for $\mathcal{N}(0, 1/d)$
  - precompute and store **Codebook**
    - Lloyd-Max centroids are denser where the PDF is tallest (near zero) and sparser in the tails

- **Step 3: Inverse Rotation**

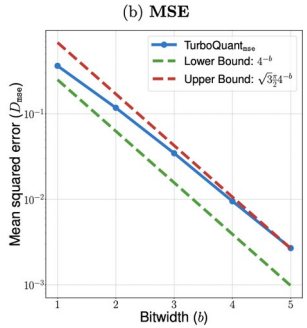
$$\hat{\mathbf{x}} = \Pi^{\top} Q_{\mathrm{LM}}(\Pi\,\mathbf{x})$$

Distortion Bounds

$$D_{\mathrm{MSE}} \leq \frac{\sqrt{3}\pi}{2} \cdot \frac{1}{4^b}$$

Lower Bound

$$D_{\texttt{mse}}(Q) := \mathbb{E}\left[\left\|\boldsymbol{x} - Q^{-1}\left(Q(\boldsymbol{x})\right)\right\|_2^2\right] \geq \frac{1}{4^b}$$

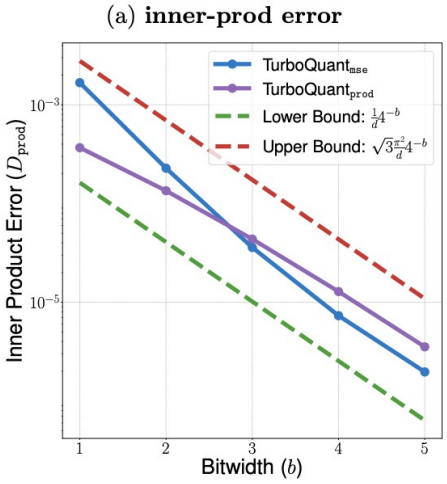


Proposed Methods (3/3)

Inner Product TurboQuant

- MSE optimized quantizers are biased for inner product
- Goal: an unbiased inner product quantizer
- Step1: apply $Q_{\texttt{mse}}$ with b-1 bit-width
- Step2: apply a $Q\mathcal{JL}$ on the residual error

$$\boldsymbol{r} := \boldsymbol{x} - Q_{\texttt{mse}}^{-1}\left(Q_{\texttt{mse}}(\boldsymbol{x})\right)$$



unbiased

$$\mathbb{E}\left[\left\langle \boldsymbol{y}, Q_{\texttt{prod}}^{-1}\left(Q_{\texttt{prod}}(\boldsymbol{x})\right)\right\rangle\right] = \langle \boldsymbol{y}, \boldsymbol{x}\rangle$$

Variance Bound

$$\texttt{Var}\left(\left\langle \boldsymbol{y}, Q_{\texttt{qjl}}^{-1}\left(Q_{\texttt{qjl}}(\boldsymbol{x})\right)\right\rangle\right) \leq \frac{\pi}{2d}\cdot\|\boldsymbol{y}\|_2^2$$

Lower Bound

$$D_{\texttt{prod}}(Q) = \mathbb{E}\left[\left|\langle \boldsymbol{y}, \boldsymbol{x}\rangle - \langle \boldsymbol{y}, Q^{-1}\left(Q(\boldsymbol{x})\right)\rangle\right|^2\right] \geq \frac{\|\boldsymbol{y}\|_2^2}{d}\cdot\frac{1}{4^b}$$

Distortion Bound

$$D_{\texttt{prod}}(Q_{\texttt{prod}}) := \mathbb{E}\left[\left|\langle \boldsymbol{y}, \boldsymbol{x}\rangle - \langle \boldsymbol{y}, Q_{\texttt{prod}}^{-1}\left(Q_{\texttt{prod}}(\boldsymbol{x})\right)\rangle\right|^2\right] \leq \frac{\sqrt{3}\pi^2\cdot\|\boldsymbol{y}\|_2^2}{d}\cdot\frac{1}{4^b} \text{ for any } b \geq 0.$$

Experiments (1/4)

Empirical Validation

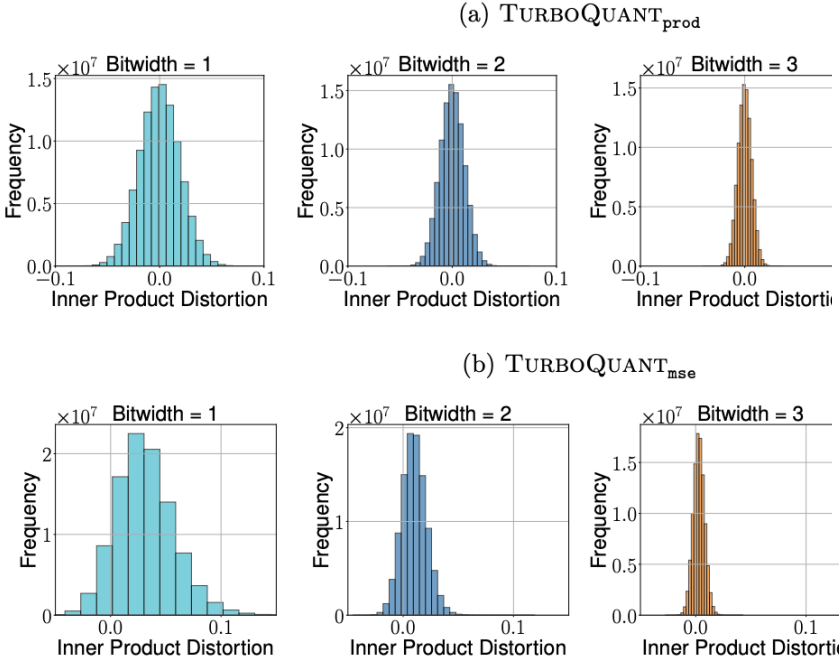


Figure 1: Error distribution of TurboQuant$_{prod}$ and TurboQuant$_{mse}$ for Inner Product Estimation.

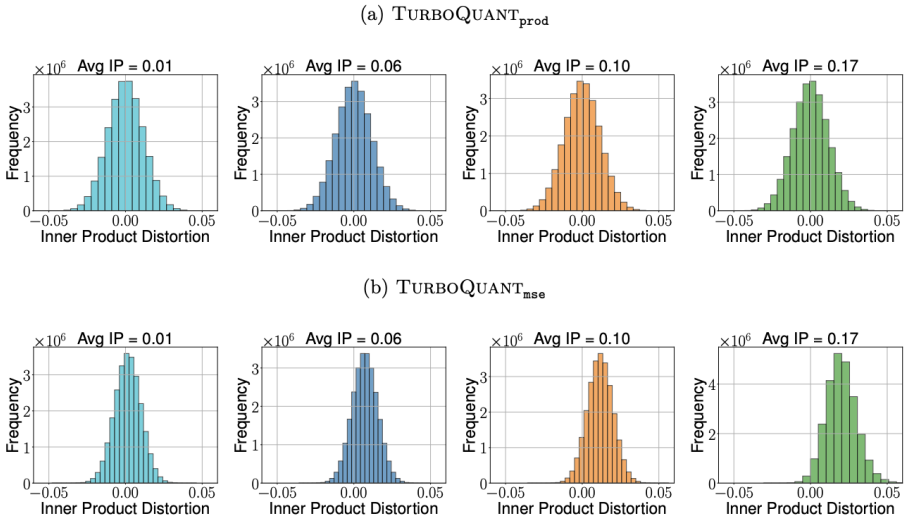


Figure 2: The variance of Inner-product error remains constant for TurboQuant$_{prod}$, while in TurboQuant$_{mse}$ increases with the average inner product. Bit-width is $b = 2$.

Experiments (2/4)

Efficacy in online KV cache quantization

- **Needle-In-A-Haystack**

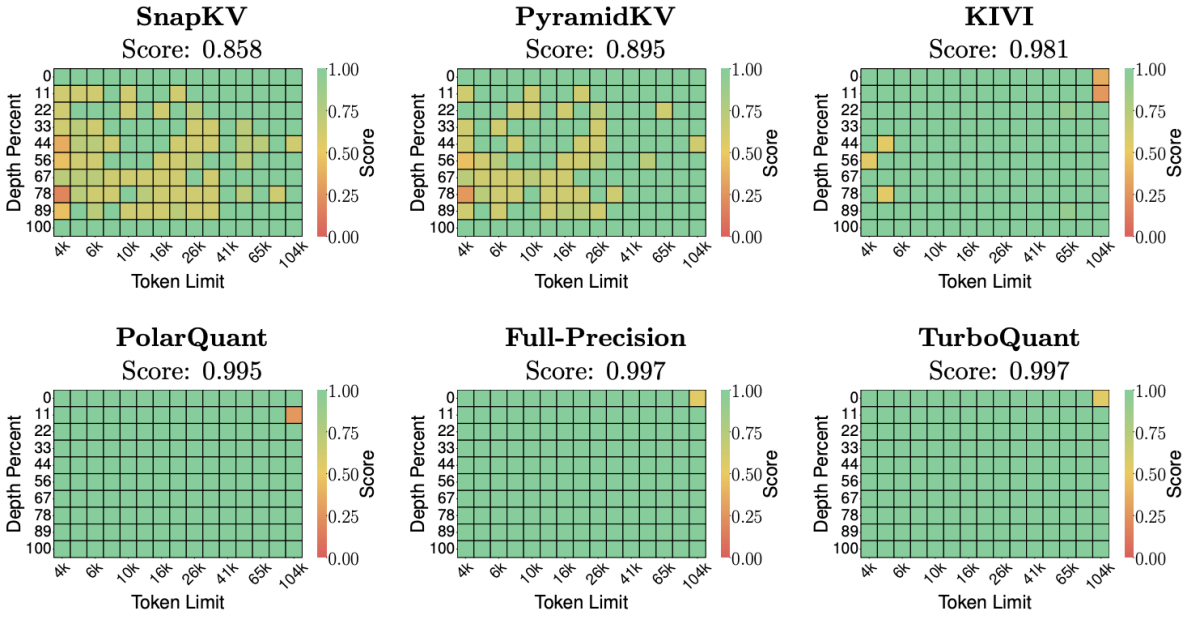


Figure 4: Evaluation of `Llama-3.1-8B-Instruct` on the "Needle-In-A-Haystack" test, where a model must retrieve a hidden sentence from long-context sequences. While some methods struggle with recall, TurboQuant, despite being more than 4× quantized, achieves the same exact performance as the uncompressed baseline.

Experiments (3/4)

Efficacy in online KV cache quantization

- **End-to-end Generation on LongBench**

Method	KV Size	SingleQA	MultiQA	Summarization	Few shot	Synthetic	Code	Average
				`Llama-3.1-8B-Instruct`				
Full Cache	16	45.29	45.16	26.55	68.38	59.54	46.28	50.06
KIVI	3	43.38	37.99	27.16	68.38	59.50	44.68	48.50
KIVI	5	45.04	45.70	26.47	68.57	59.55	46.41	50.16
PolarQuant	3.9	45.18	44.48	26.23	68.25	60.07	45.24	49.78
TurboQuant (ours)	2.5	44.16	44.96	24.80	68.01	59.65	45.76	49.44
TurboQuant (ours)	3.5	45.01	45.31	26.00	68.63	59.95	46.17	50.06
				`Ministral-7B-Instruct`				
Full Cache	16	47.53	49.06	26.09	66.83	53.50	47.90	49.89
TurboQuant (ours)	2.5	48.38	49.22	24.91	66.69	53.17	46.83	49.62

Table 1: LongBench-V1 [10] results of various KV cache compression methods on `Llama-3.1-8B-Instruct`.

Experiments (4/4)

Near Neighbour Search

Approach	d=200	d=1536	d=3072
Product Quantization	37.04	239.75	494.42
RabitQ	597.25	2267.59	3957.19
TurboQuant	0.0007	0.0013	0.0021

Table 2: Quantization time (in seconds) for different approaches across various dimensions using 4-bit quantization.

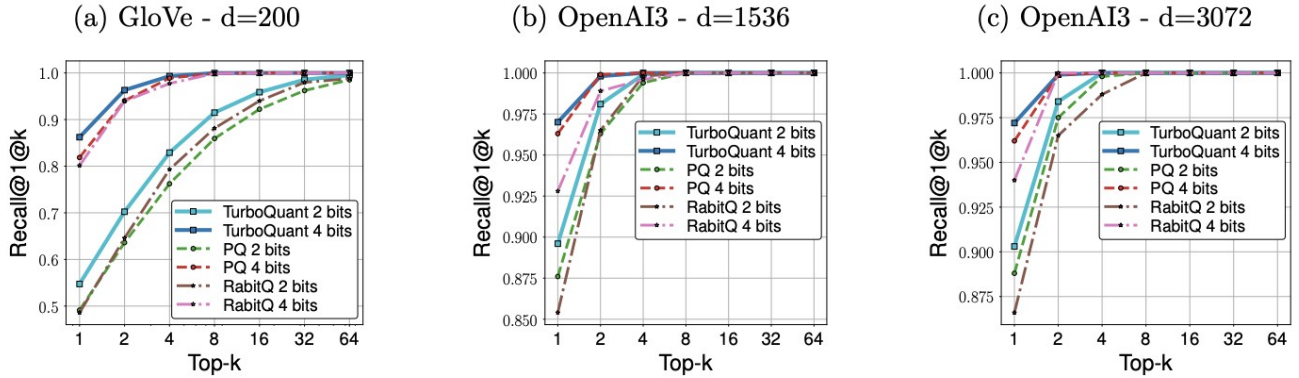


Figure 5: Recall comparison on different datasets with different embedding dimensions.

Conclusion (1/1)

conclusion

- Contribution: near-optimal, online(data-obvious) quantization
- 2 (PolarQuant, QJL),
- ,
  - Rotation, Dequantize overhead
  - edge-device ?
  - Video ?(Generation, Understanding)
    - Video KV Rotation , ?

Experiments

- https://vllm.ai/blog/2026-05-11-turboquant#accuracy-results
- https://research.nvidia.com/labs/eai/blogs/kv-cache-compression-and-its-infra-problems/
- https://arxiv.org/pdf/2604.19528

Future Work (1/1)

Further Readings

- A White Paper on Neural Network Quantization
- TURBOQUANT w/ Video Understanding
- SnapKV: LLM Knows What You are Looking for Before Generation

Thank You!