Revisiting RaBitQ and TurboQuant: A Symmetric Comparison of Methods, Theory, and Experiments

Jianyang Gao[1], Yutong Gou[2], Yuexuan Xu[2], Jifan Shi[2], Yongyi Yang[3], Shuolin Li[4], Raymond Chi-Wing Wong[5], Cheng Long[2]

[1]ETH Zurich [2]Nanyang Technological University [3]University of Michigan [4]Tsinghua University [5]Hong Kong University of Science and Technology

Methodology

Preprocessing: **random rotation / JL transformation** for both methods.

RaBitQ codebook: **uniform, shifted-integer grid** — pick rescaling factor t, round to nearest grid point, store 1 scalar.

TurboQuant codebook: **non-uniform, k-means centroids** (Lloyd–Max) - quantize to nearest centroid, apply QJL to residual for unbiased estimate.

IP estimate: RaBitQ uses **native unsigned-int arithmetic (no decoding)**; TurboQuant needs codebook lookup/decoding.

Code size: 1 scalar + D unsigned B-bit ints (TurboQuant unbiased mode: +1 scalar).

Theoretical Guarantees

RaBitQ — optimal. Matches the Alon–Klartag (FOCS 2017) space–distortion trade-off: a sub-Gaussian tail bound gives bit-width **B = Θ(log log(1/δ))**.

TurboQuant — suboptimal. Only a variance (MSE) bound is proved. Chebyshev's inequality => **B = Θ(log(1/δ))** - exponentially worse in δ.

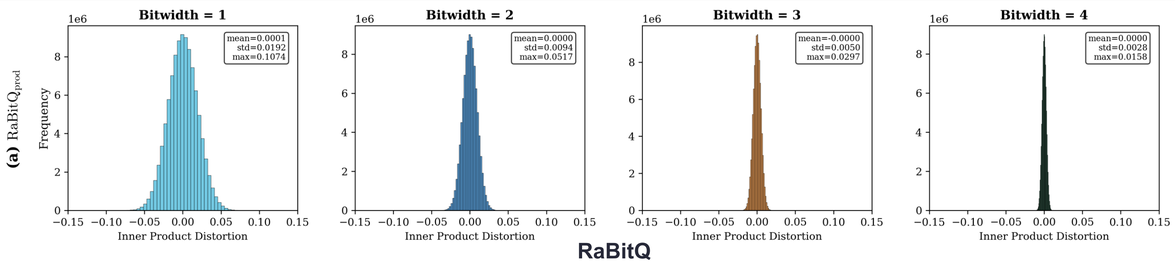
Experimental Setup

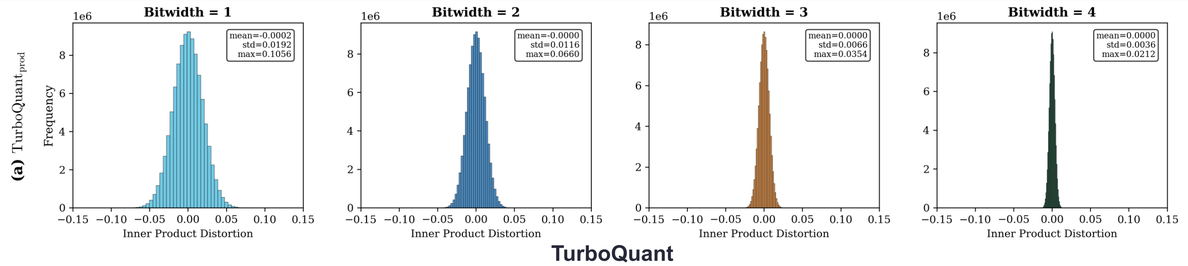
Datasets: DBpedia-1536, GloVe-200, OpenAI3-1536/3072.

Hardware: Nvidia A100 (80GB) + dual Xeon Gold 6418H (96 threads)

Code: github.com/VectorDB-NTU/rabitq-turboquant-comparison

Quantization Accuracy





TurboQuant shows no consistent advantage. For the unbiased variants, RaBitQ has smaller std. and max error across most tested bit-widths.

Quantization Efficiency (100K vectors, 4-bit, in seconds)

Approach	d=200	d=1536	d=3072
RaBitQ (CPU)	0.125	1.003	4.176
$\text{RaBitQ}_{\text{fastOn-FWHT}}$ (CPU)	0.085	0.143	0.218
RaBitQ (GPU)	0.009	0.065	0.152
$\text{RaBitQ}_{\text{fastOn-FWHT}}$ (GPU)	0.003	0.008	0.013
TurboQuant (GPU)	0.011	0.114	0.276

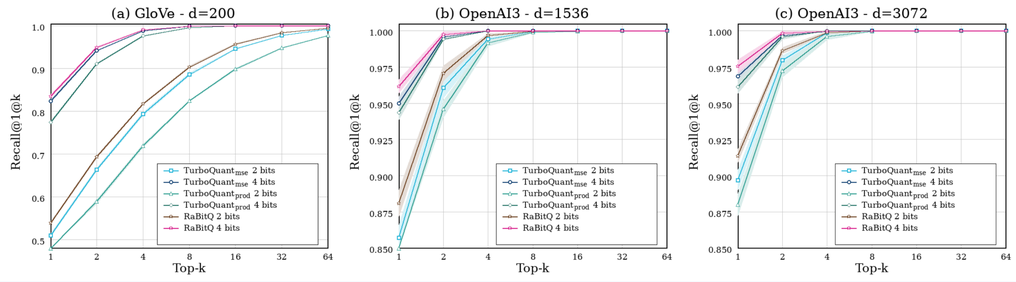
Results in TurboQuant Paper

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

- **RaBitQ on single-core CPU, multithreading disabled**
- **TurboQuant on A100 GPU**

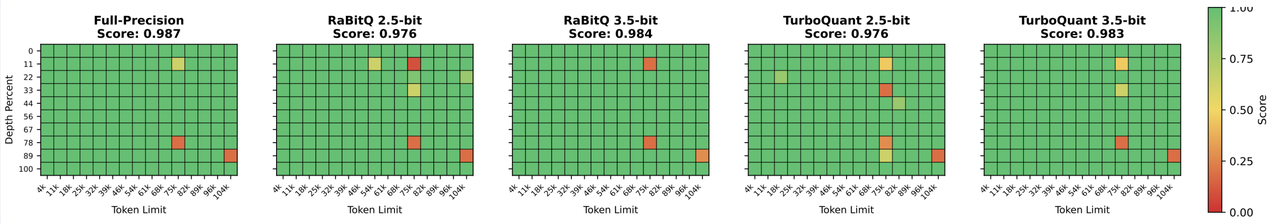
On identical GPU hardware, RaBitQ is 1.2–1.8× faster than TurboQuant across all tested dimensions.

Nearest Neighbor Search



RaBitQ consistently achieves higher recall than both TurboQuant variants across all datasets and B settings.

KV Cache Quantization (NIAH and LongBench-E)



Model	Method	SingleQA	MultiQA	Summ	Few shot	Synthetic	Code	Avg
Llama-3.1-8B	Full Cache (16-bit)	45.39	45.76	26.38	68.60	59.12	48.00	50.39
	RaBitQ 2.5-bit	**45.46**	43.92	24.25	68.08	**59.56**	46.53	49.51
	$\text{TurboQuant}_{\text{mse}}$ 2.5-bit	44.65	**45.57**	**24.70**	**68.13**	59.21	**47.10**	**49.76**
	RaBitQ 3.5-bit	44.90	44.96	25.45	68.29	**59.86**	46.57	49.87
	$\text{TurboQuant}_{\text{mse}}$ 3.5-bit	**45.67**	**46.61**	**25.92**	**68.34**	59.54	**47.67**	**50.45**
Ministral-8B	Full Cache (16-bit)	51.30	56.85	26.00	69.34	57.84	55.69	54.10
	RaBitQ 2.5-bit	**50.23**	**56.97**	**23.88**	68.80	**58.50**	53.37	**53.25**
	$\text{TurboQuant}_{\text{mse}}$ 2.5-bit	49.33	56.72	23.64	**68.96**	58.00	**54.41**	53.16

RaBitQ and TurboQuant are matched - NIAH scores differ by ≤0.002; LongBench-E overall average differs by ≤0.58 pts across all tested configurations.

Reproducibility Issues

1. RaBitQ baseline in the TurboQuant paper was run single-core CPU, multithreading disabled, in Python — vs. TurboQuant on an A100 GPU. **Not disclosed.**
2. Reported TurboQuant quantization times are ~2 orders of magnitude off from what the released code reproduces on the same GPU.
3. Reported RaBitQ recall values fall below our measured 1-std band across 10 seeded runs for ANN search.

Conclusion

1. RaBitQ **matches or beats** TurboQuant in accuracy.
2. RaBitQ is **substantially faster** in quantization.
3. RaBitQ **wins consistently** on NN search recall.
4. The two are **comparable** in KV-cache quantization.

Paper

Code