QUANTIZATION IN ANN SEARCH

Håkon Noren Myhr

February 13, 2026

Quantization Methods Tested

Two phases

1. Rotational quantization and int8 with metadata.
2. Matryoshka embeddings with binary quantization and reranking.

Rotational quantization and metadata

Summary

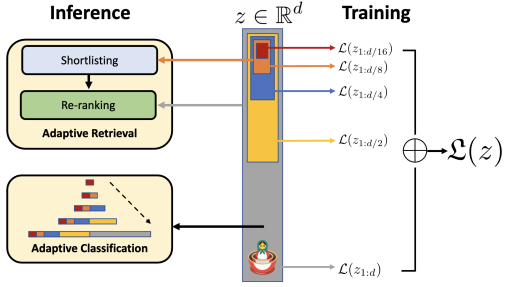
- Rotation does not seem to help much.
- Adding metadata to int8 (decoded distance) improves significantly.

	SIFT 1M		Wikipedia	
Quantization	TH10	TH100	TH10	TH100
1-bit	13.9	17.2	54.4	51.5
1-bit w/ rot	15.0	19.3	57.4	52.3
8-bit	73.7	92.1	69.3	86.3
8-bit w/ rot	70.4	92.8	70.1	86.4
8-bit SQ	**98.2**	**98.9**	99.1	**99.5**
8-bit SQ w/ rot	96.5	97.8	**99.4**	99.4

Matryoshka embeddings

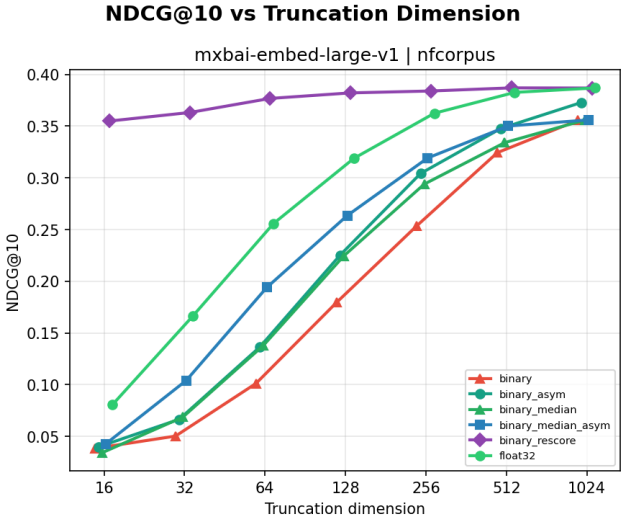
Matryoshka embeddings allows compression along two axis.

Dim	fp32	int8	binary
1024	1×	4×	32×
512	2×	8×	64×
256	4×	16×	128×
128	8×	32×	256×
64	16×	64×	512×



Nice property: $\langle x, q \rangle = \langle x_{1:d}, q_{1:d} \rangle + \langle x_{d:n}, q_{d:n} \rangle$

Matryoshka embeddings



Matryoshka embeddings

Asymmetric scoring only quantize documents:

$$\langle q,\, \hat{d} \rangle = \langle q,\, d \rangle + \underbrace{\langle q,\, \varepsilon_d \rangle}_{\text{one error term}}$$

Symmetric scoring quantizes both:

$$\langle \hat{q},\, \hat{d} \rangle = \langle q, d \rangle + \underbrace{\langle q, \varepsilon_d \rangle + \langle \varepsilon_q, d \rangle + \langle \varepsilon_q, \varepsilon_d \rangle}_{\text{three error terms}}$$

Less error with no extra storage.

Median binarization:

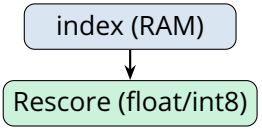
$$\hat{d}_i = \operatorname{sign}(d_i - \mu_i)$$

where $\mu_i = \operatorname{median}_j(d_i^{(j)})$.

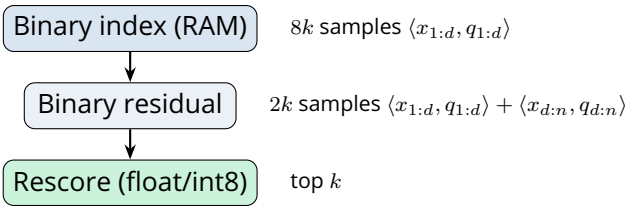
Centering at the median ensures each bit is balanced.

Matryoshka embeddings

Standard rescore:



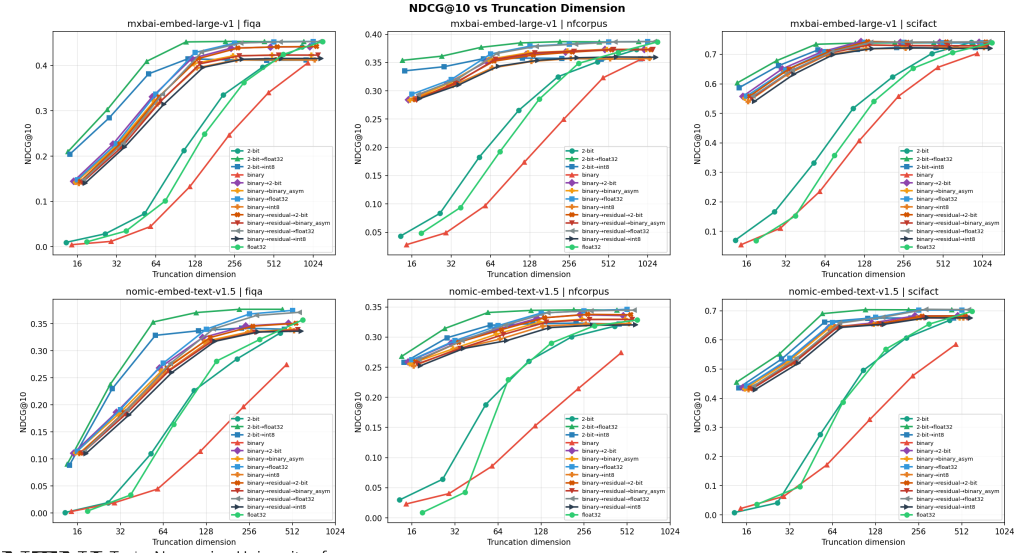
Residual rescore:



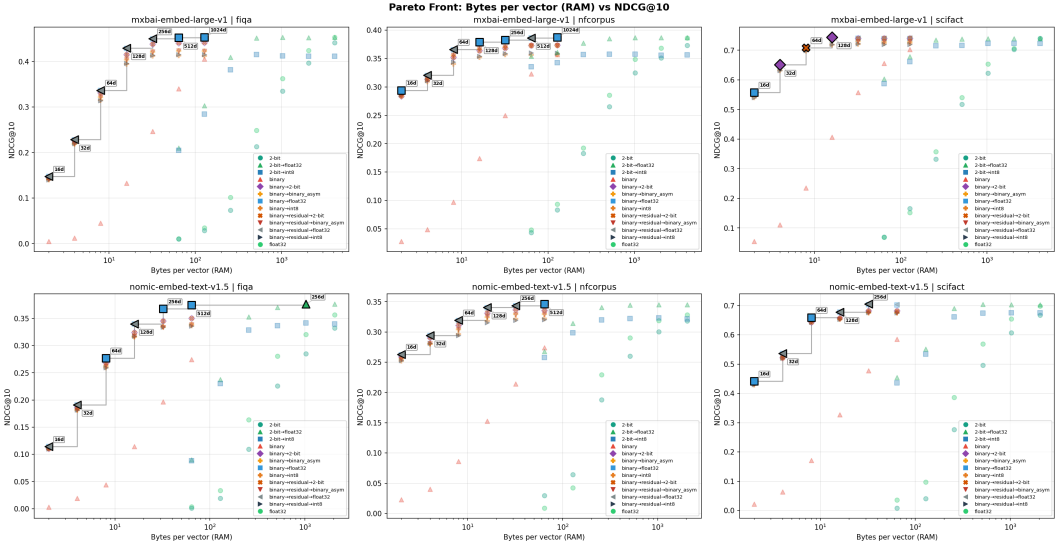
Final rescore is always asymmetric:

$$\text{score} = \langle q_{\text{float}},\ \hat{d}_{\text{quant}} \rangle$$

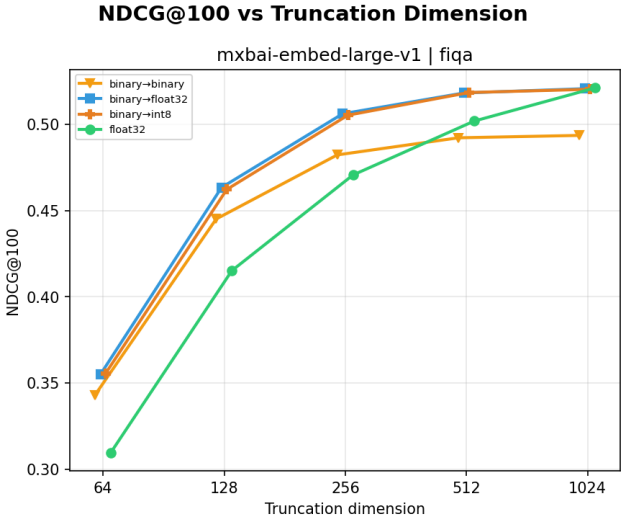
Matryoshka embeddings



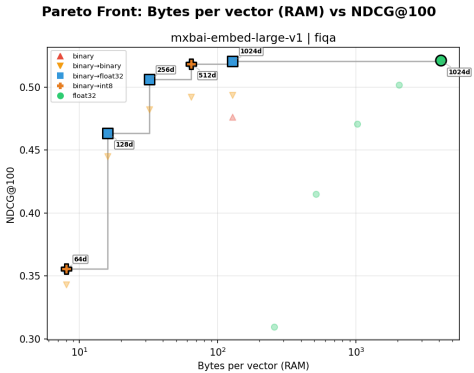
Matryoshka embeddings

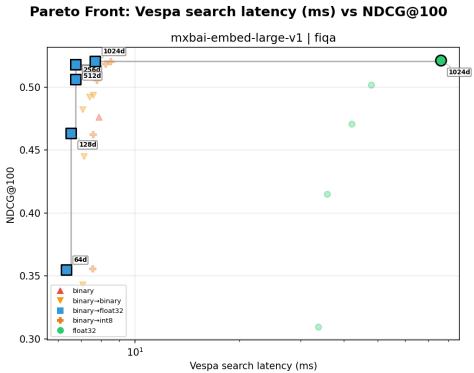


Matryoshka embeddings: vespa cloud



Matryoshka embeddings: vespa cloud





Matryoshka embeddings: vespa cloud

Binary→float32 is great!

Method	Trunc. Dim.	Compr.	Retain. (NDCG@100)	Lat. (ms)
float32†⋆	1024	1×	100.0%	75.9
binary→float32†⋆	1024	32×	99.9%	7.7
binary→float32⋆	512	64×	99.4%	6.8
binary→float32†⋆	256	128×	97.1%	6.7
binary→float32†⋆	128	256×	88.9%	6.5
binary→float32⋆	64	512×	68.1%	6.3

† Pareto-optimal (memory) ⋆ Pareto-optimal (latency)

Matryoshka embeddings: vespa cloud

Method	Trunc. Dim.	Compr.	Retain. (NDCG@10)	Lat. (ms)
float32	1024	1×	100.0%	75.9
binary→float32*	1024	32×	100.0%	7.7
binary→float32*	512	64×	99.9%	6.8
binary→float32†*	256	128×	99.2%	6.7
binary→float32†*	128	256×	94.8%	6.5
binary→float32*	64	512×	74.3%	6.3

† Pareto-optimal (memory) * Pareto-optimal (latency)

Matryoshka embeddings: vespa cloud

Method	Dim	Compr.	Retain. (NDCG@100)	Lat. (ms)
float32†⋆	1024	1×	100.0%	75.9
float32	512	2×	96.3%	48.0
float32	256	4×	90.3%	42.1
float32	128	8×	79.7%	35.7
float32	64	16×	59.4%	33.8
binary	1024	32×	91.4%	7.9
binary→binary	1024	32×	94.7%	7.6
binary→float32†⋆	1024	32×	99.9%	7.7
binary→int8	1024	32×	99.8%	8.5
binary→binary	512	64×	94.4%	7.4
binary→float32⋆	512	64×	99.4%	6.8
binary→int8†	512	64×	99.5%	8.2
binary→binary	256	128×	92.5%	7.1
binary→float32†⋆	256	128×	97.1%	6.7
binary→int8	256	128×	97.0%	7.8
binary→binary	128	256×	85.4%	7.1
binary→float32†⋆	128	256×	88.9%	6.5
binary→int8	128	256×	88.7%	7.6
binary→binary	64	512×	65.8%	7.1
binary→float32⋆	64	512×	68.1%	6.3
binary→int8†	64	512×	68.2%	7.6

† Pareto-optimal (memory) ⋆ Pareto-optimal (latency)

Compression Calculator

Find the optimal cost-accuracy trade-off for your vector search workload. Configure your setup and we'll compute the Pareto-optimal compression strategies.

1 Configure Your Workload

EMBEDDING MODEL
mxbai-embed-large-v1 (1024d)

NUMBER OF VECTORS
1,000,000

MAX EMBEDDING DIMENSION
1024

VESPA CLOUD PLAN
Commercial

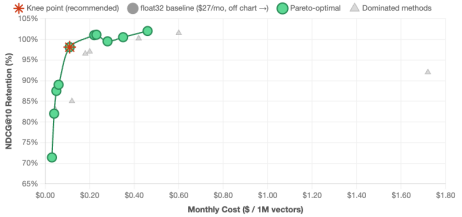
PRIORITY
Balanced (recommended)

Calculate Optimal Compression

Costs are based on Vespa Cloud pricing: RAM at $0.01/GB/hr (ANN index) and Disk at $0.0004/GB/hr (rescore vectors). RAM is 25x more expensive, making index format the dominant cost lever.

250x	$0.11	98.1%
COST REDUCTION	MONTHLY COST	QUALITY RETENTION
vs float32 baseline	per 1M vectors	NDCG@10 vs float32

Cost vs Accuracy Pareto Front — Live Simulation



Recommended Configurations — Top 5 Pareto-Optimal

#	PIPELINE	DIM	QUALITY RETENTION	COST / MO	SAVINGS	USE CASE

Compression Calculator

Find the optimal cost-accuracy trade-off for your vector search workload. Configure your setup and we'll compute the Pareto-optimal compression strategies.

250x
COST REDUCTION
vs float32 baseline

$0.11
MONTHLY COST
per 1M vectors

98.1%
QUALITY RETENTION
NDCG@10 vs float32

1 Configure Your Workload

EMBEDDING MODEL
mxbai-embed-large-v1 (1024d)

NUMBER OF VECTORS
1,000,000

MAX EMBEDDING DIMENSION
1024

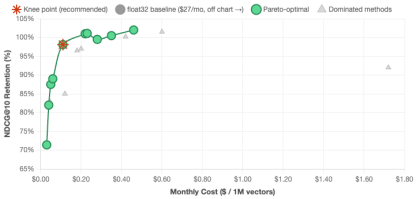
VESPA CLOUD PLAN
Commercial

PRIORITY
Balanced (recommended)

Calculate Optimal Compression

Cost vs Accuracy Pareto Front

Live Simulation



Recommended Configurations

Top 5 Pareto-Optimal

Summary

Conclusion

1. Huge potential memory savings! (w/ good embedder + resample)
2. Many tricks to regain accuracy. (asym, calibration, metadata, rotation?)

Things to investigate

1. Measure vector distribution $\rightarrow$ choose quantization scheme.
2. Make a feature for guiding customers without expensive experiments.