Low-Bit N-Gram Embeddings for Language Models

A research group paper discussion

Paper by Stephen Chung and Wenyu Du (DualverseAI)

arXiv:2610.31646v1 25 October 2026

Takeaway: use fewer bits per value to keep, or expand, n-gram table capacity.

Can precision buy more table addresses?

Question

Can hashed n-gram memories keep their modeling benefit with fewer bits per stored value?

74–75%

Table memory saved at 4 bits, with $> 99\%$ of the original loss improvement retained.

122–126%

Improvement with $4\times$ as many rows at 2 bits, while using 52–54% of baseline table memory.

47–55%

Table memory saved in two existing large models, with $< 1.2\%$ held-out perplexity change.

Paper abstract and Secs. 3.2, 3.4, 3.6; PDF pp. 4–9. All percentages concern n-gram tables.

Hash collisions make row count valuable

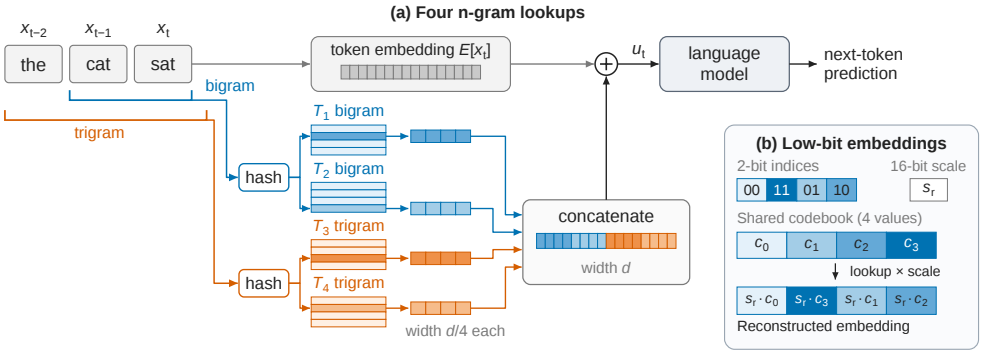
- N-gram keys are numerous; hashing maps them into a fixed set of learned rows.
- Fewer rows mean more unrelated n-grams share a whole embedding.
- Four tables, each with m rows of width $d/4$, store about md values.

Scale of the issue

The 31M Llama-family backbone has $d = 512$. At $m \approx 262$k, its n-gram tables contain about **134M values** before compression. The model name excludes these tables.

Paper, Secs. 1–2 and App. B.1, Table 1; PDF pp. 1–3, 14. The 134M count is calculated from the reported dimensions.

Four lookups augment each token representation



$$u_t = E[x_t] + \mathrm{Concat}_{j=1}^{4}\, T_j\big[h_j(x_{t-n_j+1}, \ldots, x_t)\big]$$

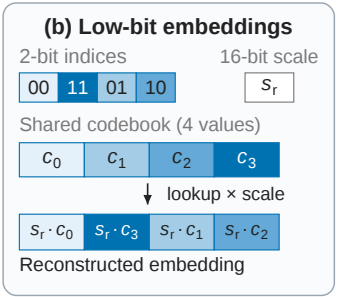
Two bigram tables and two trigram tables each return a $d/4$-wide vector.

Paper, Fig. 1, PDF p. 2; Eq. (1), PDF p. 3. Original figure.

A shared codebook plus one scale per row

$$\hat{w}_{ri} = s_r c_{q_{ri}}, \qquad q_{ri} \in \{0, \ldots, 2^b - 1\}.$$

- q_{ri}: b-bit component index.
- c_k: 16-bit value shared by a table.
- s_r: one 16-bit scale per row.



Storage per p-component row

$16p \longrightarrow pb + 16$ bits, plus a small shared codebook. For $p = 128$, 2 bits use 272 instead of 2048 bits.

Paper, Fig. 1(b), PDF p. 2; Eq. (2), PDF p. 3. Original figure panel.

Codebook and row scale solve different problems

1. **Codebook:** compare uniform, normal-quantile, and Lloyd–Max values. Lloyd–Max fits normalized components by nearest-center assignment and mean updates.
2. **Scale:** compare max-based, RMS, and error-fitted row scales.
3. **Error-fitted:** test RMS and $\alpha s_r^{\max}$ for $\alpha = 1.00, 0.95, \ldots, 0.30$; select the lowest BF16 reconstruction MSE.

Why the two levels?

The codebook captures the distribution of values across a table; s_r captures each row's magnitude.

Paper, Sec. 2 and App. A.2–A.3; PDF pp. 3, 13–14.

The baseline is the gain from adding tables

From-scratch models

Train matched models with no tables and with 16-bit tables. Quantize the trained tables and measure validation BPB (lower is better).

$$R_b = \frac{L_\varnothing - L_b}{L_\varnothing - L_{16}} \times 100\%.$$

Existing large models

Keep the backbone fixed; compress FP8 tables; measure perplexity and task accuracy against the original model.

$R_b = 100\%$ matches the 16-bit table; $R_b > 100\%$ improves on it.

Paper, Secs. 3.1, 3.4 and App. B.3–B.5; PDF pp. 4, 6, 15–16.

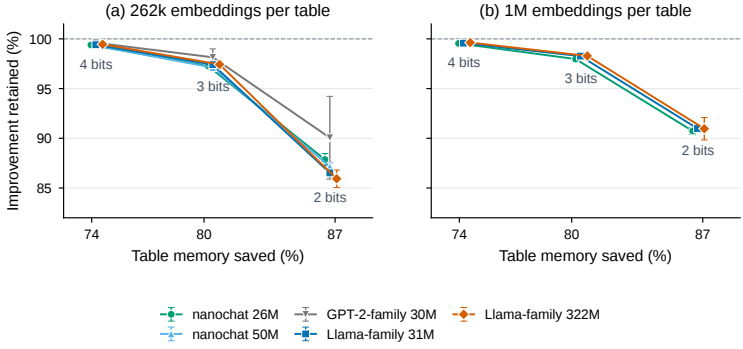
Five models, one n-gram lookup design

Model family	Backbone parameters	Width d	Rows/table
nanochat	26M	384	262k, 1M
nanochat	50M	512	262k
GPT-2	30M	512	262k
Llama	31M	512	262k, 1M
Llama	322M	960	262k, 1M

- ClimbMix training; held-out ClimbMix validation; three random seeds.
- About 500M training targets for small models, 1B for Llama 322M.

Paper, Sec. 3.1 and App. B.1–B.2, Tables 1–2; PDF pp. 4, 14–15. Parameter counts exclude tables.

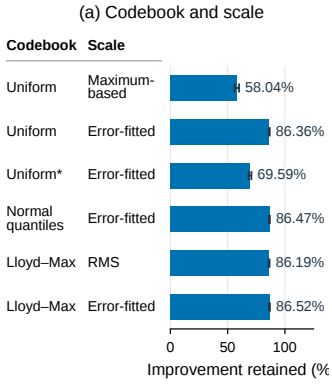
Four bits preserve nearly all of the table gain



- 4 bits: 74–75% table memory saved; 99.3–99.6% gain retained.
- 3 bits: about 80% saved; 97.1–98.1% retained.
- 2 bits: 86–87% saved; 85.9–90.1% retained.

Paper, Fig. 2 and App. C.1, Table 3; PDF pp. 5, 17. Ranges are for 262k rows/table. Original figure.

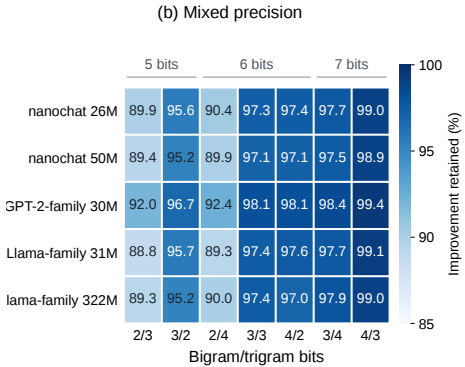
At two bits, scale fitting matters most



- Uniform + max scale: **58.04%** retained.
- Uniform + error-fitted: **86.36%**.
- With fitted scales, uniform, normal quantiles, and Lloyd–Max all lie near **86.4–86.5%**.

Paper, Fig. 3(a), PDF p. 5; 31M Llama, 262k rows/table. Original figure panel.

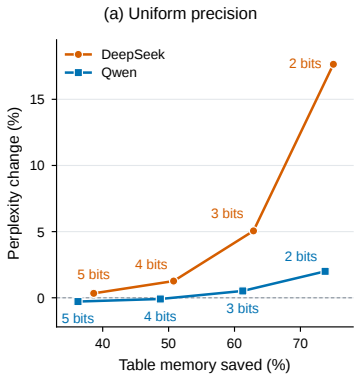
For small models, spend extra bits on bigrams



- Same total bits ⇒ approximately equal table memory.
- On 31M Llama, 4/3 bigram/trigram bits retain **99.1%**; 3/4 retain **97.7%**.
- Across five models, 4/3 retains 98.9–99.4% and saves 77–78% memory.

Paper, Fig. 3(b), PDF p. 5; full sweep in Fig. 8, PDF p. 20. Original figure panel.

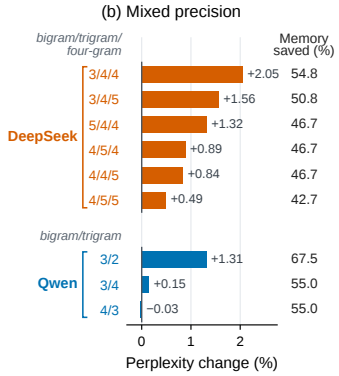
FP8 tables in two large models also compress



- Original DeepSeek-V4.1-Flash and Qwen3.8-Flash-Next tables are FP8.
- Uniform 4-bit: about half of table memory saved; validation perplexity changes by +1.26% for DeepSeek and approximately 0 for Qwen.
- Uniform 3-bit: +5.06% for DeepSeek, +0.52% for Qwen.

Paper, Sec. 3.4 and Fig. 4(a), PDF p. 6; App. B.3, PDF p. 15. Original figure panel.

Mixed precision depends on the model



- DeepSeek: 4/5/4 bits (bigram/trigram/four-gram), **46.7%** saved.
- Qwen: 4/3 bits (bigram/trigram), **55.0%** saved.
- DeepSeek favors an extra higher-order bit; Qwen favors an extra bigram bit.

Paper, Fig. 4(b), PDF p. 6; full grids in Figs. 9–10, PDF p. 21. Original figure panel.

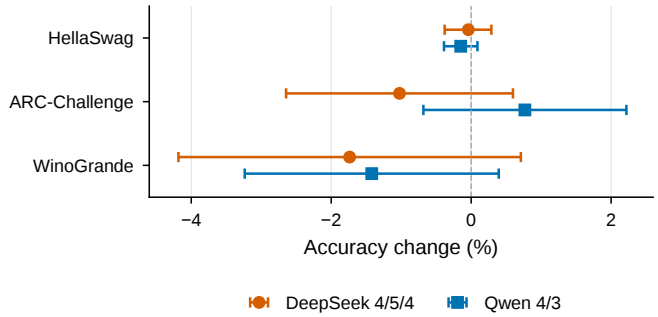
Held-out perplexity changes stay below 1.2%

Model	Corpus	Original	Low-bit	Change
DeepSeek	WikiText-2	2.7564	2.7870	+1.110%
DeepSeek	PTB	7.7730	7.7794	+0.082%
DeepSeek	C4	7.9679	7.9584	−0.119%
Qwen	WikiText-2	4.2270	4.2671	+0.949%
Qwen	PTB	4.0633	4.0558	−0.183%
Qwen	C4	9.2329	9.2354	+0.027%

DeepSeek 4/5/4 saves 46.7%; Qwen 4/3 saves 55.0% of FP8 table memory. Each corpus contributes 64 paired windows and 32,768 target tokens.

Paper, App. C.4, Table 6, PDF p. 22; evaluation protocol in App. B.3, PDF p. 15.

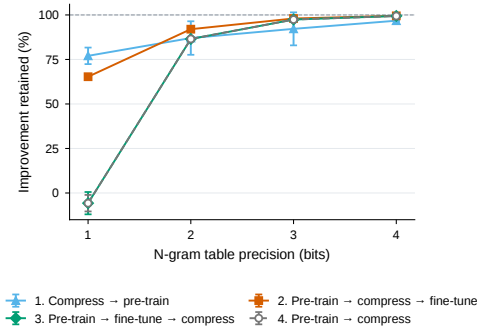
Downstream accuracy is largely preserved



- Mean of three task differences: DeepSeek −0.93 percentage points; Qwen −0.27 points.
- All six task-level paired 95% bootstrap intervals include zero.

Paper, Fig. 5, PDF p. 7; App. C.4, Table 7, PDF p. 22. HellaSwag, ARC-Challenge, WinoGrande. Original figure.

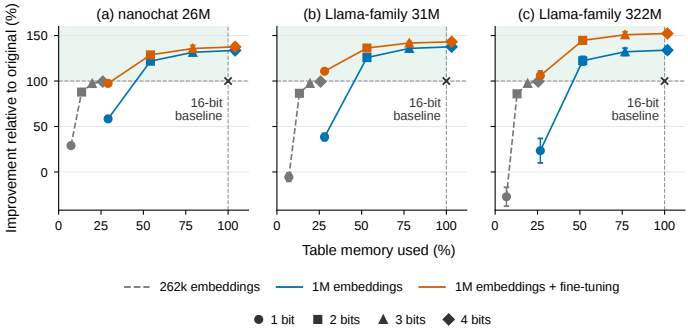
Adaptation recovers aggressive compression



- 1-bit post-compression: −5.7% retained; compress then fine-tune: **65.3%**.
- 1-bit quantized-forward pre-training: **77.0%** retained.
- Fine-tuning uses 381 extra updates for 262k-row tables, with all weights trainable.

Paper, Fig. 6, PDF p. 8; App. C.5, Table 8, PDF p. 23; App. B.4, PDF pp. 15–16. Original figure.

More rows can beat the original table at lower cost



At 1M rows/table and 2 bits: **52–54%** of the 262k-row, 16-bit table memory gives **122–126%** of its loss improvement; compress-then-fine-tune reaches **129–145%**.

Paper, Fig. 7, PDF p. 8; App. C.1, Table 4, PDF p. 18; App. C.5, Table 9, PDF p. 23. Original figure.

What the reported memory savings cover

- **Stored n-gram tables:** indices, one 16-bit scale per row, and shared codebooks; the percentage is not whole-model memory.
- **Quantized training:** a BF16 master copy remains for gradient updates, so the storage ratio is not a training peak-memory ratio.
- **Deployment:** the paper reports model quality and table storage, without end-to-end throughput or decode-cost measurements.
- **Large-model evidence:** fixed backbones; three held-out corpora and three downstream tasks.

Paper, Apps. B.3–B.5, PDF pp. 15–16; Sec. 3.4, PDF pp. 6–7.

Take-home: allocate bits and rows together

1. **4 bits** are a low-risk storage point in the five from-scratch models: about 75% less table memory with $> 99\%$ of the table gain.
2. **2 bits + more rows** can give better quality than smaller 16-bit tables at about half the table memory; fine-tune after compression.
3. **Mixed precision** should follow measured per-table sensitivity; DeepSeek and Qwen prefer different allocations.

Paper, Secs. 3.2–3.6, PDF pp. 4–9. Full paper: arXiv:2610.31646v1.

Backup: implementation of the low-bit representation

- Hash keys: $k_t^{(2)} = x_t + (V+1)x_{t-1}$; $k_t^{(3)} = x_t + (V+1)x_{t-1} + (V+1)^2 x_{t-2}$, with $V = 8192$.
- Lloyd–Max samples up to 4M normalized components/table and runs 60 assignment/update iterations.
- Quantized-forward training keeps BF16 embeddings and passes gradients through a straight-through estimator.
- Codebook values and row scales are BF16; indices are packed at the selected width.

Paper, Apps. A.1–A.3 and B.4; PDF pp. 13–16.

Backup: component-wise hashing at two bits

Model	Whole row	Same column	Fully random
nanochat 26M	91.71	93.61	93.95
GPT-2 30M	93.69	99.57	100.16
Llama 31M	87.03	100.28	98.39

Values are % of the improvement from conventional 16-bit tables. Per-component addresses improve means in these three 2-bit comparisons; the Llama same-column estimate has 10.54-point seed SD.

Paper, App. C.6, Table 10, PDF p. 24; three seeds, 262k rows/table.