Continuum Attention: Exact Retrieval over a Continuous Memory

Piecewise-linear key/value trajectories and exponential moments

Afshin Khadangi

arXiv:2610.20586v2 | 9 November 2026

Publication not found as of 2026-11-09. arXiv record
Official code: Metal and CUDA implementation

Paper identity and status checked 2026-11-09.

CA replaces point lookup with integration along memory

- Softmax attention scores a query against keys at discrete token positions.
- CA joins adjacent keys and values into continuous piecewise-linear trajectories.
- A query creates an exponential relevance density and integrates values over its visible prefix.

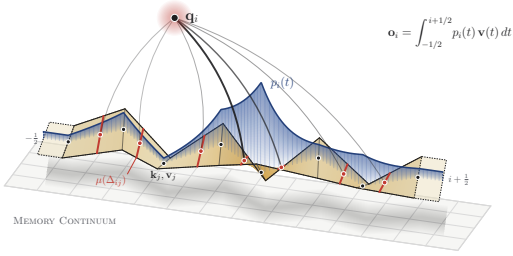
The decoder stays the same; the retrieval rule changes.

Paper, Sec. 1, PDF pp. 1–2.

Each query retrieves from a normalized continuous density

$$\mathbf{k}_j(u) = (1-u)\mathbf{k}_{j-1} + u\mathbf{k}_j, \qquad \mathbf{v}_j(u) = (1-u)\mathbf{v}_{j-1} + u\mathbf{v}_j$$

$$s_i(t) = \frac{\gamma}{\sqrt{d_h}}\mathbf{q}_i^\top \mathbf{k}(t), \quad p_i(t) = \frac{e^{s_i(t)}}{Z_i}, \quad \mathbf{o}_i = \int_{-1/2}^{i+1/2} p_i(t)\mathbf{v}(t)\,dt$$



Paper, Sec. 2.1–2.2, Eqs. (2)–(8), PDF pp. 2–3; Fig. 1, p. 1.

Affine scores make every segment integral exact

For segment j, let $a = s_i(j-1)$ and $\Delta = s_i(j) - s_i(j-1)$:

$$J_n(\Delta) = \int_0^1 u^n e^{\Delta u}\, du, \quad M_j = e^a J_0(\Delta)$$

$$\mathbf{U}_j = e^a\big[(J_0 - J_1)\mathbf{v}_{j-1} + J_1\mathbf{v}_j\big] = M_j\big[(1-\mu)\mathbf{v}_{j-1} + \mu\mathbf{v}_j\big], \quad \mu = \frac{J_1}{J_0}$$

- J_0 and J_1 are sufficient for the forward mass and value numerator.
- μ moves the effective retrieval point toward the endpoint with higher score.
- Closed forms have finite limits at $\Delta = 0$.

Paper, Sec. 2.3, Eqs. (10)–(18), PDF pp. 3–4.

Half-token caps preserve discrete attention at equal scores

$$Z_i^{\mathrm{cap}} = \tfrac{1}{2}e^{s_i(0)} + \tfrac{1}{2}e^{s_i(i)}, \quad \mathbf{U}_i^{\mathrm{cap}} = \tfrac{1}{2}e^{s_i(0)}\mathbf{v}_0 + \tfrac{1}{2}e^{s_i(i)}\mathbf{v}_i$$

$$\mathbf{o}_i = \frac{\mathbf{U}_i^{\mathrm{cap}} + \sum_{j=1}^{i} \mathbf{U}_{ij}}{Z_i^{\mathrm{cap}} + \sum_{j=1}^{i} M_{ij}}$$

- Equal visible scores give every token equal total mass: uniform causal attention.
- In general, CA is a convex combination of original values: weights are nonnegative and sum to one.

Paper, Sec. 2.4, Eqs. (19)–(22), PDF p. 5.

The analytic backward pass needs moments only through order two

- Score endpoint gradients combine adjacent segment contributions and the half-token caps.
- Query and key gradients follow from the complete endpoint-score gradients.
- Value gradients use the final normalized effective weights.
- The learned positive gain $\gamma = e^{\rho}$ has an analytic scalar gradient.

Stable evaluation

Near $\Delta = 0$, evaluate J_n with a series; elsewhere shift scores by the visible maximum before exponentiation.

Paper, Secs. 2.5–2.6, Eqs. (23)–(35), PDF pp. 5–6.

CA is exact, but it remains quadratic attention

Computation

Segment integrals are independent and can be tiled across GPU query prefixes. No quadrature or sequential scan over memory is needed.

Cost

Arithmetic is $\mathcal{O}(T^2 d_h)$, as in dense softmax. The paper's CUDA path materializes intermediate attention tiles.

The method changes the inductive bias, not the asymptotic complexity.

Paper, Sec. 2.6, PDF p. 6; Sec. 3.6, PDF pp. 10–11.

A five-seed study isolates continuous integration from temperature

- TinyStories byte LM: six layers, width 256, four heads, contexts 192 and 512.
- Eight attention settings $\times$ five paired seeds $\times$ two contexts $=$ 80 runs.
- Fixed softmax $\tau = 0.30$ ($B_{0.30}$) is parameter matched to fixed-gain CA $\tau = 0.30$ (P).
- Learned-gain controls test whether results depend on optimizing the score scale.

Paper, Appendix A, PDF p. 15; development protocol and results, pp. 16–18.

Matched-temperature CA lowers NLL at both context lengths

Configuration	$T = 192$	$T = 512$
Softmax $B_{0.30}$	1.029903	1.174585
CA P, fixed $\tau = 0.30$	**1.009888**	**1.126540**
Improvement (nats/byte)	0.020016	0.048045
Paired 95% CI	[0.013779, 0.026253]	[0.033504, 0.062587]

- CA wins all five paired seeds at both lengths.
- Learned-gain H vs. G has a CI that includes zero at 192, but not at 512.

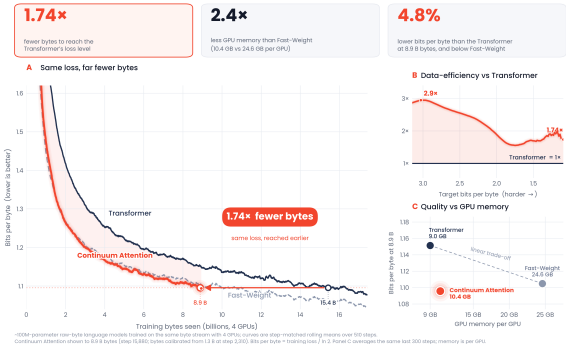
Paper, Appendix A.1 / Table A.1, PDF p. 16; paired comparisons, Table A.3, p. 18.

The larger byte-level model has a matched training recipe

- 101.2M parameters; 256-byte vocabulary; 2,048-byte context; eight layers, 16 heads.
- 26.2B FineWeb-Edu bytes, 50K steps, batch 524,288 bytes per step.
- Four NVIDIA V100 GPUs, FP16 training, three-seed evaluation.
- Baselines: Transformer, FastWeight, Byte Latent Transformer, and AU-Net.

Paper, Sec. 3.1, PDF pp. 7–8.

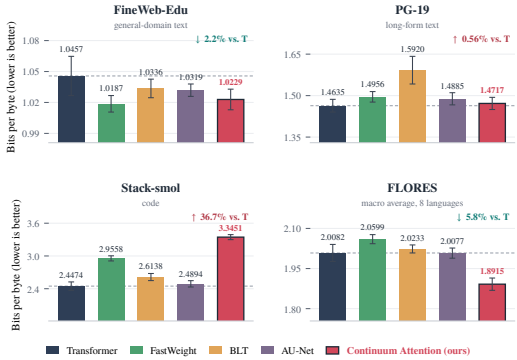
CA reaches the selected loss target with fewer training bytes



- At about 1.10 training BPB: CA uses 8.9B bytes; Transformer uses 15.4B (1.74× data efficiency).
- At 8.9B bytes, CA BPB is 4.8% lower; it uses 10.4 GB/GPU vs. FastWeight's 24.6 GB.

Paper, Sec. 3.2 / Fig. 2, PDF p. 8; threshold and memory comparisons are specific to this run.

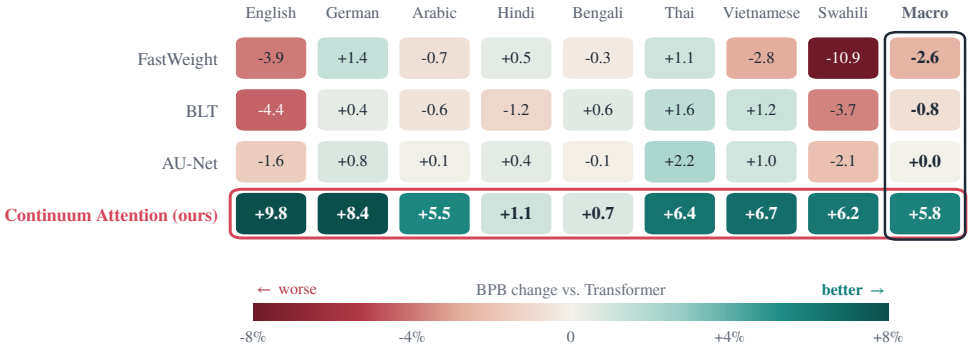
The largest language-model gain is multilingual



- FLORES macro BPB: CA 1.8915 vs. AU-Net 2.0077, averaged across three seeds.
- CA is competitive on FineWeb-Edu and PG-19, but is weakest on Stack-smol code.

Paper, Sec. 3.3 / Fig. 3 and Table 1, PDF p. 9. Lower BPB is better.

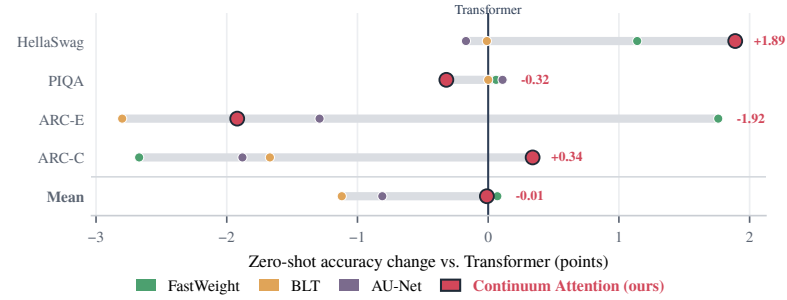
FLORES improvements are positive in all eight languages



- Macro BPB is 1.8915; relative improvement is 5.8% over the strongest macro baseline.
- Even the smallest gain, Bengali, is positive (+0.7% vs. Transformer).

Paper, Sec. 3.4 / Fig. 4 and Table 2, PDF p. 10; macro averages languages uniformly.

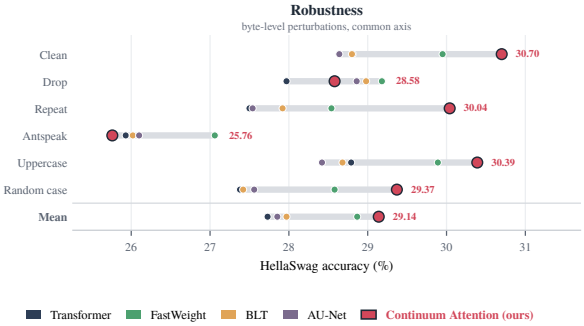
CA wins two multiple-choice tasks, not the overall mean



- Highest scores: HellaSwag 30.70% and ARC-Challenge 23.08%.
- Four-task mean: 37.01%, slightly below Transformer 37.02% and FastWeight 37.09%.

Paper, Sec. 3.5 / Fig. 5 and Table 3, PDF p. 11. Differences in the plot are percentage points.

Byte perturbation results show specific robustness trade-offs



- CA leads on clean, repeated-byte, uppercase, and random-case prompts.
- FastWeight is stronger on byte deletion and antspeak; CA falls from 30.70% clean to 25.76% antspeak.

Paper, Sec. 3.5 / Fig. 6 and Table 4, PDF pp. 11–12.

The GPU kernel is verified, while end-to-end speed remains open

- V100 isolated operator, $B = 2$, $H = 16$, $T = 2{,}048$, $d_h = 64$: 16.60 ms forward + backward.
- Comparators: 23.47 ms conventional tiled; 19.52 ms fused score/gradient in a same-run comparison.
- Causality tests at lengths through 2,048: future-token perturbations leave earlier outputs unchanged; tested future gradients are zero.
- Full-model one-V100 inference: about 7.8×10^4 input bytes/s at batch 8.

Interpretation

These are operator-level timing and verification results, not a demonstrated end-to-end training speedup.

Paper, Sec. 3.6, PDF pp. 10–11; small-model numerical preflight, Appendix A.4 / p. 18.

What CA establishes, and what the experiments leave open

Supported here

- Exact causal integration with normalized, nonnegative weights.
- Matched-temperature NLL gains in five-seed small-model tests.
- Strong FLORES BPB across all eight evaluated languages.

Current trade-offs

- Quadratic arithmetic; intermediate attention tiles are materialized.
- Lower Stack-smol code performance and uneven task results.
- Larger-model and longer-context generalization is untested.

Take-away: interpolate memory, integrate exactly, and measure the systems cost separately.

Paper, Secs. 3–5, PDF pp. 7–12; full identity: arXiv:2610.20586v2.