

TextNCA: Neural Cellular Automata for Language Modeling via Hierarchical Local Attention

Avni Mittal¹ Avinash Anand² Ashutosh Kumar¹ Dikshant Kukreja¹
Kritarth Prasad² Sushane Dulloo¹ Erik Cambria³ Timothy Liu⁴
Zhengkui Wang² Rajiv Ratn Shah⁵

¹MIDAS, IIT Kanpur, India ²Singapore Institute of Technology, Singapore

³Nanyang Technological University, Singapore ⁴NVIDIA AI Technology Centre, Singapore

⁵IIT Kanpur, India

Abstract

Neural Cellular Automata (NCAs) generate complex spatial patterns in vision by repeatedly applying a shared local rule to persistent cell states, combining local perception, weight-shared iteration, and gated in-place updates. We test this principle in language modelling with TEXTNCA, a 1D causal model in which token positions act as cells, windowed attention provides local perception, and a GRU-style gate updates their states. Its hierarchical variant cascades increasingly wide attention windows $w \in \{8, 32, 128\}$ and reuses each stage’s parameters for T_s iterations. We train roughly 30M-parameter models on SlimPajama for 60k steps and evaluate on WikiText-103. TextNCA trails a similarly sized Transformer-6L (60.3 versus 52.8 best-run PPL), so we study it as an analytical probe rather than a performance alternative. Reordering or flattening the windows raises PPL by 16.7–70.8, whereas a non-iterating sliding-window Transformer preserving the staged order performs comparably across three seeds. These results identify the narrow-to-wide schedule as the dominant factor and provide no reliable evidence that iterated weight sharing improves perplexity at this scale. Intermediate-state analyses support a prepare-then-decode account in which early stages reorganise representations and the final wide stage performs most next-token loss reduction.

1 Introduction

Neural Cellular Automata (NCAs) evolve persistent cell states by repeatedly applying a shared local rule (Mordvintsev et al., 2020; Wolfram, 2002). Their combination of local perception, weight-shared iteration, and gated in-place updates suggests that sequence context might be built progressively through local interactions. Language modelling provides a demanding test of this hypothesis because next-token prediction requires information across multiple context ranges. We investigate

whether the NCA principle transfers from vision to language and which of its properties drives the resulting behaviour.

Existing language models isolate subsets of these properties. Universal Transformers (Dehghani et al., 2019) and Relaxed Recursive Transformers (Bae et al., 2025) combine iterated weight sharing with global attention; ALBERT (Lan et al., 2020) shares parameters across layers without iteration; and sliding-window Transformers (Beltagy et al., 2020; Zaheer et al., 2020) use local attention without iteration or weight sharing. None combines all three properties in an autoregressive language model.

We introduce TEXTNCA¹, a 1D causal realisation in which token positions act as cells, windowed attention provides local perception, and a GRU-style gate updates persistent token states. Its hierarchical variant uses windows $w \in \{8, 32, 128\}$ and reuses the same parameters for T_s updates within each stage. Our motivation is twofold. First, as an inductive bias, the narrow-to-wide schedule requires local representations to be built before wider associations become available. Our results identify this schedule as the only strongly supported part of the NCA prior: iterated weight sharing has at most a small, bounded effect whose sign we cannot resolve at this scale. Second, reusing one in-place update within each stage exposes a step-by-step dynamical process under a fixed transformation, allowing us to track hidden states, update magnitudes, cosine similarity, and GRU gates across iterations. The tools we use, including the logit lens, linear probing, CKA, and attention maps, also apply to Transformers; our narrower claim is that weight-shared iteration gives these measurements a recurrent trajectory to describe. Because matched Transformer baselines achieve lower perplexity, we

¹Code is available at <https://github.com/AvniMittal13/TextNCA>.

treat TextNCA as an analytical test of the NCA hypothesis rather than a performance alternative or a generally more interpretable model. Our contributions are:

1. **A controlled probe of NCA-inspired language modelling.** We map the NCA primitive to a causal language model and isolate seven design axes: perception kernel, gate, step embedding, orchestration, stage count K , window schedule, and iteration count T_s .
2. **The narrow-to-wide schedule is the dominant architectural factor.** Perturbing the schedule costs tens of PPL, while a non-iterating control preserving its window order performs comparably and has the better three-seed mean. A wide-first control that still ends wide costs +21.9 PPL, supporting the full progression rather than either endpoint.
3. **Iteration exposes a recurrent trajectory, but its perplexity effect is unresolved.** A from-scratch T_s sweep is U-shaped, and matched controls show no reliable perplexity gain from iterated weight sharing at this scale. Intermediate-state and distance-controlled analyses support a prepare-then-decode account in which early stages reorganise representations and the final wide stage performs most next-token loss reduction.
4. **Randomising the training iteration count enables inference at several counts.** One model remains stable at $T_s \in \{2, 4, 6\}$, but reaches 101 PPL versus 60.3 for the fixed-iteration flagship.

2 Related Work

Iterated and weight-shared transformers. Universal Transformers (Dehghani et al., 2019) apply a shared block for T iterations with *global* attention and a stateless residual update; under our protocol, UT reaches PPL 91.7 vs. Hier-TextNCA’s 60.3, indicating weight sharing under global attention is not what drives Hier-TextNCA. ALBERT (Lan et al., 2020) shares parameters across layers without iteration, and we evaluate an ALBERT-style autoregressive model in §5.2; Relaxed Recursive Transformers (Bae et al., 2025) share layer blocks with low-rank relaxations and retain global attention; looped-transformer work

(Geiping et al., 2025; Saunshi et al., 2025) explores test-time iteration depth at much larger scales. Our T_s U-shape at 30M is independent evidence for the reading that iteration in LMs behaves as effective depth, and not as a refinement axis. Earlier, deeply iterative architectures (Neural GPU (Kaiser and Sutskever, 2016), DEQ (Bai et al., 2019)) do not target autoregressive LM.

Local, sparse, and efficient attention. Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2020) combine sliding-window attention with sparse global tokens; sliding-window transformers without iteration or weight sharing are the family of our SWin-TF-Staged control. State-space and long-convolution models (Mamba (Gu and Dao, 2024), Hyena (Poli et al., 2023)) replace attention entirely; we train Mamba under our protocol in §5.2. We tested GLA (Yang et al., 2024) as a perception kernel in the hierarchical body and found that despite GLA beating local softmax at single scale (129 vs. 140 PPL at 60k steps), the hierarchical body inverts this ordering (72.0 vs. 60.3), a non-trivial kernel-schedule interaction.

NCA and position of this work. NCAs were introduced for image generation (Mordvintsev et al., 2020) and extended to texture synthesis (Mordvintsev et al., 2021; Palm et al., 2022); subsequent work introduced hierarchical NCA variants for medical image segmentation, using multi-scale rules to propagate global information across the image (Kalkhof et al., 2023; Mittal et al., 2025). Concurrent work uses NCA-generated data to pre-train transformers (Lee et al., 2026). TextNCA is the first work, to our knowledge, to use NCA-style computation as the language-model architecture. None of the prior architectures above combines all three NCA properties simultaneously; this paper is not novel because it uses iteration, locality, or weight sharing alone, but because it tests their conjunction under a hierarchical staged schedule and isolates which part actually drives the empirical behaviour.

3 TextNCA: Architecture

TEXTNCA implements the NCA primitive as repeated weight-shared updates to a sequence of token states; Table 1 maps the image-domain NCA components onto our realisation. Each of the three

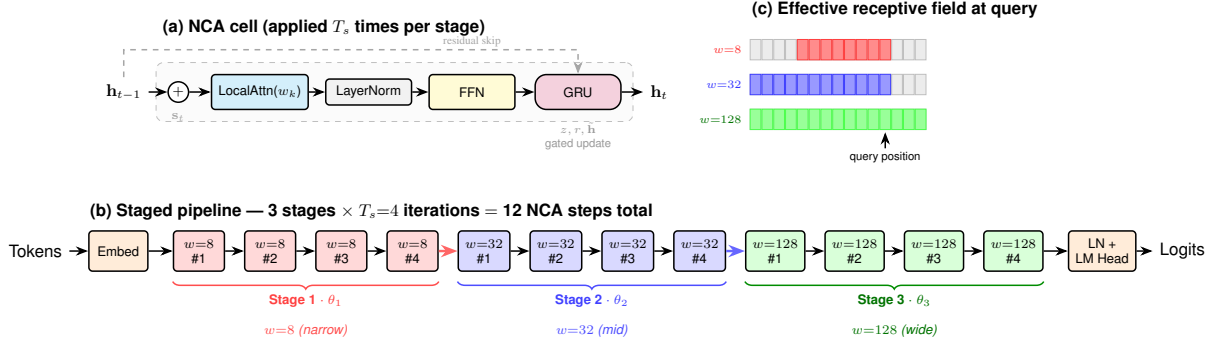


Figure 1: **Hierarchical TextNCA architecture.** (a) One NCA update combines local attention, LayerNorm, an FFN, and a GRU-gated residual update. A learned step embedding conditions the shared rule across iterations. (b) Three stages with windows $w \in \{8, 32, 128\}$ each apply their own parameters for $T_s=4$ shared-weight iterations (12 updates total). (c) Additive receptive-field upper bound for a query near the end of a 15-token sequence, progressing from nearby tokens to the full sequence. In practice, retrieval remains bounded by the active stage’s window (§6).

stages applies the same local-attention update T_s times under its own parameter set, so the model is a hierarchical NCA. TextNCA is, to our knowledge, the first language model to combine strictly local attention, iterated weight sharing, and a GRU-style gated in-place update. We use “NCA” as a structural description of the per-stage rule and treat the schedule of window sizes across stages as the architectural choice that carries the empirical work. Appendix A.1 situates the primitive against the image-domain NCA of Mordvintsev et al. (2020), and Appendix A gives the full architecture specification.

NCA property	TextNCA realisation
Cell state vector	Token hidden state $\mathbf{h}_i \in \mathbb{R}^d$
Local perception kernel	Causal windowed attention, window w
Shared update rule	Tied weights within each stage
Iterative in-place update	$\mathbf{h} \leftarrow \text{Gate}(\mathbf{h}, f(\mathbf{h}))$, T_s times
Gated residual (“fire rate”)	GRU update/reset gates
Multi-scale extension	Stage cascade $w \in \{8, 32, 128\}$

Table 1: NCA primitives mapped to TEXTNCA. The defining NCA conjunction (locality, iterated weight sharing, gated in-place update) holds inside each stage; the full model is a hierarchical NCA with rule piecewise-constant across stages.

3.1 Core update

Given input tokens $\mathbf{x} = (x_1, \dots, x_L)$, we initialize token states as

$$\mathbf{h}_0 = \text{TokenEmbed}(\mathbf{x}) + \text{PosEmbed}(\mathbf{x}) \in \mathbb{R}^{B \times L \times d} \quad (1)$$

The model then applies the same recurrent update for T iterations:

$$\mathbf{h}_t = \text{Gate}(\mathbf{h}_{t-1}, \text{FFN}(\text{LN}(\text{Perceive}(\mathbf{h}_{t-1} + \mathbf{s}_t)))) \quad (2)$$

where $\mathbf{s}_t \in \mathbb{R}^d$ is a learnable step embedding. The perception module, FFN, gate, and normalization parameters are *shared across all iterations*. Final predictions are produced by

$$\text{logits} = \text{LMHead}(\text{LN}(\mathbf{h}_T)). \quad (3)$$

In all main experiments, $\text{Perceive}(\cdot)$ is local causal softmax attention with window size w , $\text{Attn}(Q_i, K, V)$:

$$\text{softmax} \left(\frac{Q_i K_{[i-w+1:i]}^\top}{\sqrt{d_h}} \right) V_{[i-w+1:i]} \quad (4)$$

The alternative perception kernels are used only for the ablation in Appendix A.3: local linear attention with a Taylor-2 feature map $\phi(\mathbf{x}) = [1, \mathbf{x}, \mathbf{x} \otimes \mathbf{x}/\sqrt{2}]$, gated linear attention following Yang et al. (2024), and causal depthwise convolution with kernel size w . The gate is a GRU-style update that combines the perception output $\mathbf{u}$ with the previous state $\mathbf{h}_{t-1}$. The update equations are in Appendix A.4, which also replaces the GRU with sigmoid, highway, residual or overwrite updates and yields the +16.3 PPL GRU-vs-residual penalty reported in §5.

3.2 Hierarchical staged TextNCA

The hierarchical variant processes tokens through K sequential stages with strictly increasing window sizes. Each stage k has its own parameter set $\theta_k = (\text{Perceive}_{w_k}, \text{LN}_k, \text{FFN}_k, \text{Gate}_k)$ and iterates T_s times with θ_k held fixed. The output of stage k

becomes the input to stage $k+1$ with no reset. The window schedule is $[w_1, w_2, w_3] = [8, 32, 128]$. For each stage $k = \{1..K\}$, the hidden state is refined T_s times:

For $t = 1..T_s$:

$$\mathbf{u} = \text{FFN}_{\theta_k}(\text{LN}_{\theta_k}(\text{Perceive}_{\theta_k, w_k}(\mathbf{h} + \mathbf{s}_{(k-1)T_s+t}))) \quad (5)$$

$$\mathbf{h} \leftarrow \text{Gate}_{\theta_k}(\mathbf{h}, \mathbf{u}). \quad (6)$$

We select $K = 3$, $T_s = 4$ (so total NCA steps = 12), and the schedule $[8, 32, 128]$ from an explicit architecture search over window schedules, stage counts, and iteration counts (§5.1–§5.3). This is the best observed configuration, not a statistically resolved optimum, because the alternative search points are single runs. Across each axis, we hold the training recipe fixed so that comparisons isolate architectural choices rather than differences in hyperparameter tuning (Appendix I).

Receptive-field profile. For causal local attention with window w_k applied T_s times in stage k , the receptive field at position i grows additively: $R_K = 1 + \sum_{k=1}^K T_s(w_k - 1)$. For $T_s=4$ and $w \in \{8, 32, 128\}$, this gives $R_1=29, R_2=153, R_3=661$, a strict narrow-to-wide cascade. This is an upper bound on what the stack could reach. In practice retrieval tracks the per-stage window instead: §6 shows that associations up to about 31 tokens appear in the $w=32$ stage, associations of 32 to 127 tokens appear only in the $w=128$ stage, and those from 135 to 260 never form (§6).

Staging vs. interleaving. Whether to expose the three window sizes *as stages* (consecutive iterations at $w=8$, then $w=32$, then $w=128$) versus *within each iteration* (cycle $[w=8 \rightarrow w=32 \rightarrow w=128 \rightarrow \text{Gate}]$ each step) makes a large difference. The staged design reaches 60.3 PPL while the interleaved variant reaches 94.4 PPL ($\Delta = 34.1$, Table 2). Alongside the schedule-perturbation ablations, this provides separate evidence that performance depends on the *ordering* of window sizes across iterations, not merely on the set of windows used.

4 Experimental Setup

4.1 Training protocol and compute

We use full backpropagation through all T steps with gradient checkpointing (Chen et al., 2016) and

AdamW (learning rate 6×10^{-4} , $\beta = (0.9, 0.95)$, weight decay 0.1, gradient clipping at 1.0), with a cosine schedule and 2000 warmup steps in bfloat16. All main-comparison models train for 60k steps on a 500M-token subset of SlimPajama (Soboleva et al., 2023), tokenised with the Mistral 32K BPE vocabulary at sequence length 512. Hier-TextNCA runs at roughly $2 \times$ the FLOPs and $2.7 \times$ lower throughput than Transformer-6L per training step, because the 12 shared updates cannot be fused like independent layers; we therefore report both matched-parameter and matched-compute comparisons (§5.2, Appendix H). Per-run FLOPs, throughput and GPU-hour figures are in Appendix J.

4.2 Language-modelling evaluation

We evaluate periodically on the WikiText-103 test split (Merity et al., 2017) and select the checkpoint with the lowest loss; the reported token-level perplexities therefore use this split for both model selection and evaluation. Because training uses SlimPajama with a Mistral 32K BPE vocabulary rather than the in-domain WikiText-103 vocabulary, our perplexities are not comparable to literature numbers that use adaptive softmax on in-domain data. We therefore restrict interpretation to internal comparisons under the same protocol; full comparison details are in Appendix I. Across the repeated 60k runs, Transformer-6L remains ahead at every seed, while the Hier-TextNCA/SWin-TF-Staged ordering reverses between the best runs and three-seed means (§5.2).

4.3 Downstream tasks

We fine-tune the available pretrained checkpoints on three classification benchmarks and one extractive QA benchmark.

Datasets and splits. IMDb (Maas et al., 2011): binary sentiment, 25k train / 25k test. AG News (Zhang et al., 2015): 4-class topic, 120k train / 7.6k test. SST-2 (Socher et al., 2013): binary sentiment, ≈ 67 k train / 872 dev. SQuAD v1.1 (Rajpurkar et al., 2016): extractive QA, ≈ 87 k train / 10.6k dev. For SST-2 and SQuAD v1.1, whose test labels are not public, we evaluate on the official dev split following standard practice.

Fine-tuning. We fine-tune each pre-trained checkpoint on each task with AdamW for 5 epochs (classification) or 3 epochs (SQuAD), reusing the LM sequence length and tokeniser. Full hyperpa-

rameters (learning rates, batch sizes, head architecture) are in Appendix I.

Metrics and seeds. Classification: top-1 accuracy on the held-out evaluation split, reported as a mean over three seeds (42, 1337, 2024). SQuAD: token-level F1 and exact-match on the dev split, computed with the official evaluator; single-seed.

4.4 Inference and mechanistic evaluations

Using frozen checkpoints, we measure answer-token perplexity on 1,500 selected ELI5 training examples and top-5 associative recall over 24,875 repeated-token dependencies in WikiText-103. A controlled copy task uses 256 prompts per distance at sequence length 288 to measure top- k recall and the first stage whose attention selects the antecedent. We also analyse intermediate states and stage-skipping, test recovery after injecting Gaussian noise of 0.25–2.0 hidden-state standard deviations, and vary the inference iteration count; full protocols are in Appendices E–G.

4.5 Baselines

Our 60k-step baselines include **Transformer-6L** and **Transformer-12L** as standard-architecture references at two depths, and **Universal Transformer** (Dehghani et al., 2019) to isolate weight sharing under global attention. **SWin-TF-Staged** preserves the narrow-to-wide window order using six distinct layers, removing iteration and weight sharing; **SWin-TF-6L** controls for window-locality without staging. At 30k-step snapshots, we additionally compare with the official HuggingFace **Mamba** implementation and an autoregressive **ALBERT-style** model that shares one Transformer block across layers. Full configurations are in Appendix A, Table 11.

5 Results

We report language-modelling perplexity on WikiText-103 (§5.1–§5.3), downstream transfer to classification and extractive QA (§5.5), and long-context diagnostics (§5.6); the experimental protocol is described in §4.

5.1 Main Finding: Schedule and Iteration

Table 2 reports the schedule and orchestration ablations, and Figure 2 summarises the main comparisons. We read the effect along three axes:

(1) **Window-schedule direction and shape.**

Model / variant	Params	tok/s	PPL
<i>Main models and controls:</i>			
Transformer-12L	54.5M	103K	44.7
Transformer-6L	35.5M	300K	52.8 (56.5 $\pm$ 3.3)
Hier-TextNCA [8, 32, 128] $\times T_s=4$	30.8M	112K	60.3 (69.6 $\pm$ 8.3)
SWin-TF-Staged [8, 32, 128] (6L)	35.5M	295K	64.5 (65.0 $\pm$ 0.9)
SWin-TF-6L (uniform $w=128$)	35.5M	298K	63.2
Universal Transformer	21.4M	151K	91.7
Hier-TextNCA (Residual gate)	26.1M	135K	76.6
Hier-TextNCA (GLA kernel)	30.9M	65K	72.0
<i>TextNCA schedule and orchestration axis:</i>			
Interleaved [8, 32, 128] $\times T=4$	27.7M	–	94.4
Reversed [128, 32, 8]	30.8M	–	131.1
Uniform [32, 32, 32]	30.8M	–	119.7
Wide-first [32, 8, 128]	30.8M	–	82.2
Non-monotone [8, 128, 32]	30.8M	–	77.0
<i>Stage-count axis (NCA steps $\approx$ 12):</i>			
$K=2$ [16, 128] $\times T_s=6$	26.1M	–	75.8
$K=4$ [4, 16, 64, 128] $\times T_s=3$	35.6M	–	71.8

Table 2: WikiText-103 PPL and training throughput for the main models and schedule/orchestration ablations at 60k steps. For repeated models, PPL reports the best run (three-seed mean $\pm$ sd); entries without parentheses are single runs. Dashes mark throughput not reported for an ablation. Reordered TextNCA schedules hold the windows, parameter count, and additive receptive-field bound fixed; the interleaved variant instead cycles the windows through one shared cell.

Varying only the schedule gives forward [8, 32, 128] at 60.3 PPL, wide-first [32, 8, 128] at 82.2 (+21.9), non-monotone [8, 128, 32] at 77.0 (+16.7), uniform [32, 32, 32] at 119.7 (+59.4) and reversed [128, 32, 8] at 131.1 (+70.8). Ending wide is insufficient: [32, 8, 128] shares the flagship’s final stage and additive bound ($R_3=661$) but loses 21.9 PPL. Dropping the narrow start (+21.9) also costs slightly more than dropping the wide end (+16.7). Together, these results support a monotonic narrow-to-wide order, rather than total context or either endpoint alone.

(2) **Stage count.** At matched total NCA steps, $K=2$ ([16, 128] with $T_s=6$) reaches 75.8 PPL and $K=4$ ([4, 16, 64, 128] with $T_s=3$) reaches 71.8. These single runs form the stage-count axis of the architecture search; we therefore report $K=3$ as best observed, not statistically resolved.

(3) **Stage-aware non-iterating controls.** SWin-TF-Staged uses six distinct layers with windows [8, 8, 32, 32, 128, 128], preserving the narrow-to-wide order while removing iteration and weight sharing. It reaches 64.5 PPL, 4.2 behind the best Hier-TextNCA run, while the uniform- $w=128$ control reaches 63.2, a gap of 2.9. Both best-run differences are smaller than

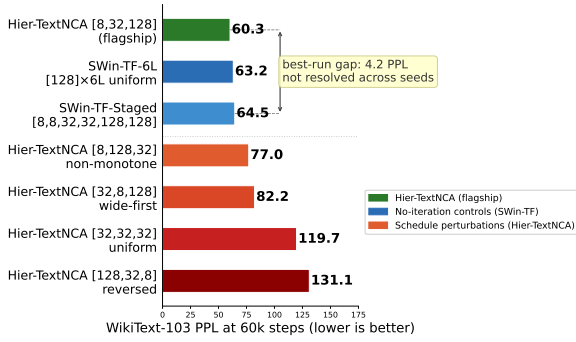


Figure 2: Best-run WikiText-103 PPL for the flagship, non-iterating controls, and schedule perturbations at 60k steps. The wide-first schedule retains the final $w=128$ stage but incurs +21.9 PPL. The bracket compares the flagship with the stage-matched non-iterating control; their 4.2-PPL best-run difference is not resolved across three seeds.

the observed variation across Hier-TextNCA runs, and SWin-TF-Staged performs better on average across three runs (65.0 vs. 69.6). We therefore find no reliable perplexity benefit from iterated weight sharing. Separately, the Universal Transformer reaches 91.7 PPL, showing that repeated weight sharing under global attention does not reproduce Hier-TextNCA’s performance. Together with the much larger schedule-perturbation penalties, these results identify window order as the stronger factor within Hier-TextNCA.

5.2 Comparison to Transformers

Table 2 reports the matched-step model comparison; training-step trajectories for the same runs are in Appendix D (Figure 7).

Seed variability. Over three seeds, Hier-TextNCA reaches 69.6 ± 8.3 PPL, compared with 65.0 ± 0.9 for SWin-TF-Staged and 56.5 ± 3.3 for Transformer-6L (Appendix B, Table 12). The 4.2-PPL best-run difference between Hier-TextNCA and SWin-TF-Staged is smaller than Hier-TextNCA’s 8.3-PPL run-to-run standard deviation, and the ordering reverses at the mean (65.0 vs. 69.6). We therefore do not claim that iteration improves perplexity. Transformer-6L remains better at every seed, while the much larger schedule-perturbation penalties remain well beyond this variation.

Parameter-sharing and efficient baselines. At 30k steps, Hier-TextNCA reaches lower PPL than Mamba, the ALBERT-style model and the Universal Transformer, while both dense transformers remain better (Table 3). The ALBERT-style base-

Model	Class	Params	PPL
Transformer-12L	dense	54.5M	56.4
Transformer-6L	dense	35.5M	64.7
Hier-TextNCA	local, iterated,	30.8M	72.1
Mamba	shared selective SSM cross-layer	~30M	88.3
ALBERT-style	weight sharing global, iterated,	19.8M	93.2
Universal Transformer	shared	21.4M	100.4

Table 3: WikiText-103 PPL at 30k training steps. Mamba uses the official HuggingFace implementation.

line is autoregressive and shares one Transformer block across all layers, following ALBERT’s cross-layer parameter sharing. Original ALBERT is a bidirectional masked-language-model encoder and therefore cannot be evaluated directly as a causal language model.

The matched-step comparison is the most favourable framing for TextNCA; under matched compute, the gap widens substantially. Hier-TextNCA also runs at $1.9\times$ the FLOPs and $2.7\times$ lower throughput than Transformer-6L per training step (Table 2). At FLOP-matched compute, Hier-TextNCA reaches PPL 70.6 versus Transformer-6L’s 52.8; at wall-clock-matched compute, 80.1 vs. 52.8 (Appendix H, Figure 20). TextNCA is not computationally Pareto-superior to transformers at this scale; any “parameter efficiency” reading should be restricted to matched-parameter, matched-step protocols.

Additional comparisons. The Universal Transformer baseline (21.4M parameters, global attention, weight sharing, stateless residual update) reaches 91.7 PPL, substantially worse than Hier-TextNCA and worse than every non-iterating sliding-window variant. Weight sharing under global attention is therefore not the architectural ingredient driving Hier-TextNCA’s behaviour; the staged local schedule is. Substituting the GLA perception kernel into the hierarchical body raises PPL from 60.3 to 72.0 (+11.7), despite GLA outperforming local softmax at single scale (129 vs. 140 PPL at 60k steps). In a short matched-budget comparison, reversing the [8, 32, 128] schedule incurs different penalties for softmax and GLA (+112.4 vs. +62.2 PPL), providing direct evidence of a kernel–schedule interaction (Appendix A.3, Table 7).

Task	TF-6L@60k	TF-12L@60k	Hier-TextNCA	SWin-TF($w=128$)	Hier-Residual	Hier-NoStep	Hier-SinStep	UT(L5)
<i>Classification (% , 3-seed mean):</i>								
IMDb	90.84	91.00	91.00	90.26	89.82	90.07	87.92	89.25
AG News	93.43	93.82	93.59	93.58	93.34	93.37	92.25	92.99
SST-2	87.16	87.50	89.00	86.70	86.47	86.35	84.06	83.94
Clf Avg	90.48	90.77	91.20	90.18	89.88	89.93	88.08	88.73
<i>Extractive QA (SQuAD v1.1):</i>								
F1	28.6	33.2	26.3	22.4	23.6	22.8	19.4	27.4
EM	8.1	9.5	7.1	5.8	6.3	5.9	4.9	7.4
Params	35.5M	54.5M	30.8M	35.5M	26.1M	30.8M	30.8M	21.4M
LM PPL	52.8 (56.5 $\pm$ 3.3)	44.7	60.3 (69.6 $\pm$ 8.3)	63.2	76.6	82.7	119.5	91.7

Table 4: Downstream evaluation at 60k pre-training steps. Classification is the three-seed mean, while SQuAD is a single-seed evaluation. For repeated models, LM PPL reports the best run (three-seed mean $\pm$ sd); other LM entries are single runs.

5.3 Ablations of Architectural Components

Having established the importance of the window schedule, we next ablate the gate, step embeddings, iteration count, and perception kernel.

Gating. Replacing the GRU with a plain residual update gives 76.6 PPL instead of 60.3, and the residual variant is also worse downstream (Table 4). This supports the GRU within the iterated model, although the PPL comparison is a single run. Pure sigmoid gating diverges; SigmoidDropout stabilizes but underperforms (71.7). More details are in Appendix A.4, Table 8.

Step embeddings. Removing the learned per-iteration embeddings s_t raises PPL from 60.3 to 82.7, while replacing them with sinusoidal embeddings raises it to 119.5 (Table 9). Learned per-step conditioning therefore substantially improves the iterated model, while the tested sinusoidal alternative performs poorly.

Iteration count T_s . We retrain Hier-TextNCA from scratch at $T_s \in \{2, 4, 6, 8\}$ with matched everything else; total NCA steps are $3T_s$. PPL is 80.3, 60.3, 73.9 and 117.9, respectively, giving a U-shaped curve with the best observed result at $T_s=4$. Because each setting is a single run, we do not interpret the smaller differences among $T_s \in \{2, 4, 6\}$. Three observations remain: (i) $T_s=8$ (24 steps) is worse than $T_s=2$ (6 steps) by 37.6 PPL, so more iterations do not monotonically improve perplexity. This resembles the degradation of image patterns when vision NCAs are run for too many steps (Mordvintsev et al., 2020). (ii) Parameter count remains essentially fixed ($\Delta < 0.2\%$), so the differences reflect the number of recurrent updates rather than model size. (iii)

The update magnitude remains nonzero at the trained horizon ($\|\Delta \mathbf{h}\| \approx 4$; Appendix E.7), rather than approaching a fixed point. Together with the test-time analysis in §7, these results are consistent with T_s acting as an effective-depth knob (Saunshi et al., 2025; Bae et al., 2025).

Perception kernel. The kernel ranking changes between single-scale and staged models, as summarized above. Full 60k and matched 30k comparisons are in Appendix A.3.

5.4 Recovery from hidden-state corruption

We apply a one-time hidden-state corruption at TextNCA’s first update and, as a control, at Transformer-6L’s second layer (Appendix E.13, Table 14). After perturbations of 0.25 and 0.5 state standard deviations, TextNCA’s remaining updates remove 98% and 90% of the induced output loss; recovery falls to 66% and 31% under stronger perturbations. Transformer-6L’s final loss remains elevated after the rest of its forward pass. This experiment measures output-loss recovery for one corruption family, using one checkpoint and injection location per architecture; it does not establish reconstruction of the clean hidden state.

5.5 Downstream Evaluation

On **classification**, Hier-TextNCA averages 91.2% across the three tasks, compared with 90.5% for Transformer-6L and 90.8% for Transformer-12L. The architecture is therefore competitive on these tasks, but the small differences do not establish a classification advantage for Hier-TextNCA.

On **extractive QA**, Hier-TextNCA reaches 26.3 F1 on SQuAD, compared with 28.6 for the similarly sized Transformer-6L (35.5M vs. 30.8M parameters) and 33.2 for the larger Transformer-12L

(54.5M). It therefore remains within 2.3 F1 of the matched-size reference, while the gap to the larger model is wider. Across the full table, local-attention variants reach 19.4–26.3 F1 and global-attention models reach 27.4–33.2 F1.

5.6 Long-context diagnostics

An inference-only ELI5 check shows no broad degradation on the selected long-form samples; the full protocol, results, and qualifications are in Appendix F.3, Table 18. Distance-stratified associative recall on WikiText-103 provides a more direct test of retrieval (Appendix F). In the 32–127 bucket, Hier-TextNCA remains close to the global-attention baselines, reaching 0.70 recall compared with 0.71 for Transformer-6L and 0.70 for Transformer-12L. In the 128–255 bucket, it falls to 0.40 while both dense Transformers retain 0.66. The iterated Universal Transformer retains 0.61 with global attention, whereas the non-iterating SWin-TF-Staged model is slightly better than Hier-TextNCA in all three in-window buckets but drops similarly beyond the widest window. Taken together, the ELI5 and recall results localise an important limitation to retrieval beyond the active window rather than a general inability to use context. This pattern is consistent with locality contributing to the SQuAD gap, but does not establish that receptive field explains the entire difference.

Appending one global-attention block at the readout does not resolve the QA gap: the 34.0M-parameter Hier-Hybrid variant reaches 22.4 F1 (Appendix C, Figure 6). This negative result applies to the tested readout modification; it does not establish where global attention would need to be added.

6 Interpretability Analysis

We analyse how hidden states change across the repeated updates and summarise the main findings here; additional loss, attention, gate, and logit-lens analyses are reported in Appendix E.

Topic information is most linearly decodable in stage 2. The middle stage gives the highest topic-probe accuracy (Figure 3). The final stage slightly reduces this linear separability while preparing the representation for next-token prediction.

Loss reduction is concentrated in stage 3. Applying the final LM head to each intermediate state (Appendix E.1, Figure 8; 64 WikiText-103 sequences, 32,768 tokens) shows that $S3.2 \rightarrow S3.3 \rightarrow$

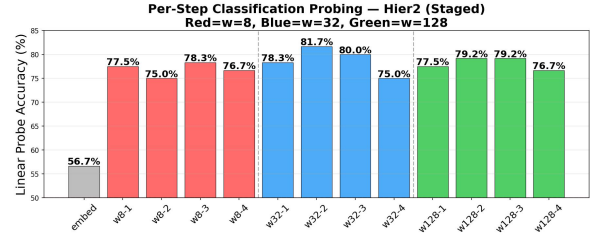


Figure 3: Per-step accuracy of a 5-way topic classifier trained on the hidden state at each NCA step (embedding shown as step 0). Accuracy rises from 56.7% at the embedding to 77.5% after the $w=8$ stage, peaks at 81.7% during $w=32$, and ends at 79.2% after $w=128$.

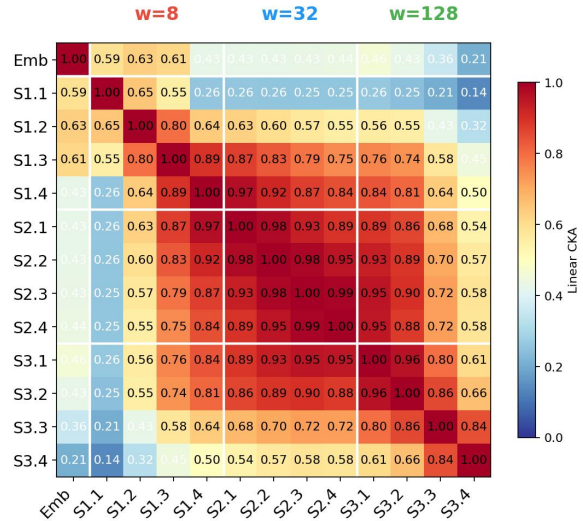


Figure 4: Linear CKA between the embedding state and all 12 NCA steps (2,048 WikiText-103 tokens; random-pair floor ≈ 0.20). White lines mark stage transitions. S1.1 and S3.4 are the only strong outliers; states S1.4–S3.1 form a continuous cluster with pairwise CKA > 0.85 .

S3.4 reduce loss by $2.4 + 1.9 + 1.5 = 5.8$ nats. Stages 1–2 instead show per-step changes between -0.3 and $+0.2$ nats. Under this readout, topic decodability can improve without a corresponding reduction in next-token loss.

Representation change is concentrated near the endpoints. The CKA structure (Figure 4; Kornblith et al., 2019) indicates large changes near the beginning and end of the computation, with a comparatively stable middle region. The stage boundaries therefore do not correspond to complete representational resets.

Distance controls the retrieval stage. In a synthetic copy task, we compute at each update the fraction of prompts whose head-averaged query-attention argmax selects either member of the earlier cue–target pair. We say that an association

forms in the first stage in which this hit rate reaches 0.15 (256 prompts per tested distance; sequence length 288). A dependency at $d=31$ forms in the $w=32$ stage, while $d=32$ forms only in the $w=128$ stage. Retrieval remains possible at $d=127$ (top-1 recall 0.45) but falls to zero at the next tested distance, $d=135$. The natural-text replication over 24,875 repeated content-token dependencies shows the same transition (Appendix F.2, Table 17).

The stages have different roles. A dependency at $d=20$ lies inside stage 1’s additive receptive-field bound ($R_1=29$) but forms only in stage 2, so direct attention association follows the per-stage window rather than this accumulated bound. Stage 1 performs little direct retrieval: among the tested distances, only $d=3$ forms there, while the geometrically reachable $d=5$ and $d=7$ cases form in stage 2. For every reachable distance in the synthetic task, the target enters the top five only in stage 3.

Prepare, then decode. Together, these results support a *prepare-then-decode* account. Stages 1 and 2 reorganise the representation, while stage 3 converts it into next-token predictions. The stage-skipping intervention provides additional support: skipping stage 3 is the most damaging condition, while running stage 3 directly from the embeddings also performs poorly (Appendix E.12, Table 13). Because these interventions produce off-distribution hidden states, we interpret their ordering rather than the absolute perplexities.

7 Test-time iteration-count control

The iterated design supports inference with a different iteration count T'_s without retraining. For the deterministic flagship, however, performance is best at the training value $T'_s=4$ and degrades past 10^3 PPL for $T'_s \in \{1, 8, 10, 12\}$. Smaller values truncate the trained update sequence, while values above four introduce unseen step-conditioning values.

Training with sinusoidal step embeddings and $T_s \sim \text{Uniform}\{2, 4, 6\}$ maintains PPL between 101 and 108 across the trained iteration counts and partly generalises to $T'_s=5$ (PPL 164). The deterministic SinStep counterpart exceeds 10^3 PPL whenever $T'_s \neq 4$. This flexibility increases absolute perplexity: the best STOCHASTICT result is 101 PPL, compared with 60.3 for the deterministic flagship. This gap is consistent with the poor performance of sinusoidal step embeddings under fixed- T_s training, although the contribution of stochastic

training is not isolated (Appendix G). We therefore describe the result as inference-time iteration-count control rather than test-time compute scaling.

8 Conclusion

TextNCA functions as a language model but remains behind the Transformer baselines, with a larger gap under matched compute. Its main architectural result is the stage order: the monotonic narrow-to-wide schedule matters more than iterated weight sharing, which gives no reliable perplexity improvement over the matched non-iterating control. However, under the tested intervention, the remaining iterations return the output loss close to the clean baseline after mild-to-moderate hidden-state corruption. The internal analyses support a prepare-then-decode account in which stages 1–2 reorganise the representation and stage 3 produces the next-token distribution. Stochastic iteration-count training permits inference at several trained iteration counts, but increases absolute perplexity. We hope future work can improve TextNCA’s quality and efficiency toward lightweight language models whose recurrent computation supports mechanistic analysis.

9 Limitations

Experiments are limited to roughly 30M parameters, so whether the schedule and iteration findings persist at larger scales remains open (Geiping et al., 2025); we also do not report a parameter–PPL frontier. Matched steps favour TextNCA, while FLOP- and wall-clock-matched comparisons widen the Transformer-6L gap, precluding a compute-Pareto claim (Appendix H). Because training uses SlimPajama with the Mistral 32K tokenizer and evaluation uses WikiText-103, PPL is comparable only within this protocol (§4.2).

Only Hier-TextNCA, SWin-TF-Staged and Transformer-6L were repeated across seeds; other ablations are single runs, and classification differences are inconclusive (Appendix B). Mamba and ALBERT-style results use 30k steps. ELI5 uses selected training examples within 512 tokens and measures answer-token PPL, not answer quality; retrieval beyond 128 tokens may contribute to, but does not fully explain, the SQuAD gap (Appendix F.3). The global-readout result does not identify where global context is needed. Iteration-count control costs 101 vs. 60.3 PPL, and the U-shaped sweeps do not support monotonic refine-

ment. The corruption probe covers one family, checkpoint, and injection site per architecture and measures output-loss recovery, not hidden-state reconstruction.

10 Ethical Considerations

This architectural study uses models of at most 55M parameters and involves no human subjects, new data collection, or deployed system. It nevertheless inherits standard language-model risks from its datasets, particularly the web-derived SlimPajama corpus, including social biases, offensive content, personal information, and memorisation. We do not perform dedicated bias, toxicity, privacy, memorisation, or misuse evaluations; the released checkpoints should therefore not be used in consequential applications without further assessment. Training consumed approximately 80 GPU-hours on two NVIDIA RTX PRO 6000 Blackwell GPUs (Appendix J); we do not estimate carbon emissions. The release will document dataset terms and the license and intended use of each artifact (Appendix I).

Acknowledgments

Dr. Rajiv Ratn Shah is partly supported by the Infosys Center for AI at Indraprastha Institute of Information Technology, Delhi.

References

- Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. 2025. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. In *International Conference on Learning Representations*.
- Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2019. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.
- Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*.
- Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations*.
- Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. 2025. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*.
- Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling (COLM)*.
- Lukasz Kaiser and Ilya Sutskever. 2016. title=Neural GPUs learn algorithms. In *International Conference on Learning Representations*.
- John Kalkhof, Camila González, and Anirban Mukhopadhyay. 2023. Med-NCA: Robust and lightweight segmentation with neural cellular automata. In *International Conference on Information Processing in Medical Imaging (IPMI)*, pages 705–716. Springer.
- Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International Conference on Machine Learning*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.
- Dan Lee, Seungwook Han, Akarsh Kumar, and Pulkit Agrawal. 2026. Training language models via neural cellular automata. *arXiv preprint arXiv:2603.10055*.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 142–150.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.
- Avni Mittal, John Kalkhof, Anirban Mukhopadhyay, and Arnav Bhavsar. 2025. MedSegDiffNCA: Diffusion models with neural cellular automata for skin lesion segmentation. In *2025 IEEE 38th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 35–40. IEEE.
- Alexander Mordvintsev, Eyvind Niklasson, and Ettore Randazzo. 2021. Texture generation with neural cellular automata. *arXiv preprint arXiv:2105.07299*.
- Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. 2020. Growing neural cellular automata. *Distill*, 5(2).
- Rasmus Berg Palm, Miguel Gonzalez-Duque, Shyam Sudhakaran, and Sebastian Risi. 2022. Variational neural cellular automata. In *International Conference on Learning Representations*.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. 2025. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. <https://huggingface.co/datasets/cerebras/SlimPajama-627B>.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642.

Stephen Wolfram. 2002. *A New Kind of Science*. Wolfram Media.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2024. Gated linear attention transformers with hardware-efficient training. In *International Conference on Machine Learning*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NeurIPS)*.

A Full Architecture Specification

This appendix provides the detailed update equations, perception-kernel and gate variants, and non-hierarchical orchestration strategies referenced in §3.

A.1 The NCA primitive in context

A Neural Cellular Automaton is a grid of cells, each holding a state vector and updated synchronously by a shared local rule. In the image variant of

Mordvintsev et al. (2020), each cell sees a 3×3 neighbourhood through a depthwise convolution, passes the perception output through a small MLP, and writes the result back through a stochastic “fire rate” that gates the residual update. Three properties define this computation. Weight sharing: all cells use the same parameters at every step. Locality: each cell only sees its immediate neighbours. Iteration: the same rule is applied T times so that global structure emerges from repeated local interactions (Mordvintsev et al., 2020; Wolfram, 2002).

TextNCA preserves these three properties for autoregressive language modelling. Each token position i holds a state $\mathbf{h}_i \in \mathbb{R}^d$; the 2D convolution is replaced by 1D causal windowed attention with window w ; the fire rate is replaced by a GRU update gate (§A.4). The per-iteration update is in-place and shared across iterations within a stage. The mapping to image-domain primitives is summarised in Table 1. The hierarchical variant introduces three stages with their own parameter sets, so the rule is piecewise constant across stages while still being a shared local rule within each stage; this is what we mean when we describe the model as a hierarchical NCA rather than a single monolithic CA.

A.2 Core update equations

Given input tokens $\mathbf{x} = (x_1, \dots, x_L)$, initial states are

$$\mathbf{h}_0 = \text{TokenEmbed}(\mathbf{x}) + \text{PosEmbed}(\mathbf{x}) \in \mathbb{R}^{B \times L \times d}. \quad (7)$$

At each iteration $t \in \{1, \dots, T\}$,

$$\mathbf{h}_t = \text{Gate}(\mathbf{h}_{t-1}, \text{FFN}(\text{LN}(\text{Perceive}(\mathbf{h}_{t-1} + \mathbf{s}_t))))), \quad (8)$$

where $\mathbf{s}_t \in \mathbb{R}^d$ is a learnable step embedding and all parameters (Perceive, FFN, Gate, LN) are shared across all T iterations. Final predictions are

$$\text{logits} = \text{LMHead}(\text{LN}(\mathbf{h}_T)). \quad (9)$$

In the hierarchical staged variant the per-iteration sharing holds *within each stage*; the rule changes at stage boundaries. For each stage $k = 1..K$, the hidden state is refined T_s times:

For $t = 1..T_s$:

$$\mathbf{u} = \text{FFN}_{\theta_k}(\text{LN}_{\theta_k}(\text{Perceive}_{\theta_k, w_k}(\mathbf{h} + \mathbf{s}_{(k-1)T_s+t}))) \quad (10)$$

$$\mathbf{h} \leftarrow \text{Gate}_{\theta_k}(\mathbf{h}, \mathbf{u}). \quad (11)$$

The output of stage k becomes the input to stage $k+1$ with no reset.

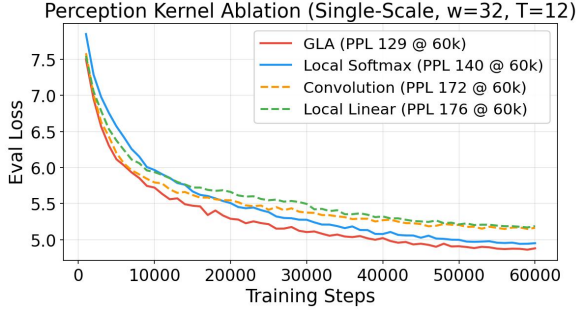


Figure 5: Perception-kernel ablation at single scale ($w=32$, $T=12$), all kernels clipped to 60k training steps. GLA (129 PPL) clearly beats local softmax (140), while convolution (172) and local linear (176) lag by 32–36 PPL. The hierarchical body inverts this ranking, with local softmax reaching 60.3 PPL and GLA 72.0.

A.3 Perception kernels

All main experiments use local softmax attention; three other kernels are used in the perception-kernel ablation (§5.3).

Local softmax attention (main). Standard multi-head attention restricted to a causal window of size w :

$$\text{Attn}(Q_i, K, V) = \text{softmax}\left(\frac{Q_i K_{[i-w+1:i]}^\top}{\sqrt{d_h}}\right) V_{[i-w+1:i]}. \quad (12)$$

Local linear attention. Approximates softmax via a Taylor-2 feature map $\phi(\mathbf{x}) = [1, \mathbf{x}, \mathbf{x} \otimes \mathbf{x}/\sqrt{2}]$, applied within the local window.

Gated linear attention (GLA). Following Yang et al. (2024), introduces data-dependent decay gates per head:

$$\mathbf{g}_j^{(h)} = \sigma(\mathbf{W}_g^{(h)} \mathbf{K}_j), \quad \text{decay}_{j \rightarrow i} = \prod_{s=j+1}^i (1 - \mathbf{g}_s^{(h)}). \quad (13)$$

Convolution. 1D causal depthwise convolution with kernel size w .

Single-scale results. At single scale ($w=32$, $T=12$) the kernel ordering on WikiText-103 PPL at 60k steps is GLA (129) < local softmax (140) < convolution (172) < local linear (176). Within the hierarchical body the ordering inverts: GLA reaches 72.0 PPL versus local softmax’s 60.3. The interaction is consistent with GLA’s data-dependent decay gates providing the largest benefit at small receptive fields where softmax attends uniformly; at $w \in \{32, 128\}$ softmax already achieves sharp query-dependent attention, and GLA’s scalar-decay bottleneck becomes a limitation.

Staged results for all four kernels. Table 5 compares all four kernels at single scale and under the staged schedule, using matched 30k-step snapshots.

Kernel	Single-scale	Staged
softmax	195.7	72.1
GLA	180.0	91.5
convolution	216.1	187.1
linear (Taylor-2)	243.8	205

Table 5: All four perception kernels at single scale ($w=32$) and staged [8, 32, 128], both at a 30k-step snapshot.

Lower perplexity is better. At single scale, GLA performs best at 180.0 PPL, followed by softmax at 195.7, convolution at 216.1, and linear attention at 243.8. Under the staged schedule, softmax becomes the strongest kernel at 72.1 PPL, ahead of GLA at 91.5. Convolution and linear attention improve under staging but remain the two weakest kernels, at 187.1 and 205 PPL. Thus, staging does not preserve the single-scale ranking: it reverses the order of the two leading kernels while leaving the lower two unchanged.

Per-stage decoding. Applying the LM head at every NCA step on WikiText-103 shows that the two leading kernels decode on different schedules rather than differing by a constant offset. GLA starts reducing loss earlier, already at $w=32$, while softmax stays near the uniform-vocabulary ceiling until $w=128$ and then reaches a lower final loss.

Stage	softmax CE	GLA CE
S1 ($w=8$)	9.90	10.06
S2 ($w=32$)	10.16	8.68
S3 ($w=128$)	6.73	5.82
Final step	4.10	4.28

Table 6: Per-stage next-token cross-entropy in nats for the staged softmax and staged GLA models.

Kernel by schedule. To test whether schedule sensitivity depends on the kernel, we train a matched 2×2 grid of {softmax, GLA} and {forward [8, 32, 128], reversed [128, 32, 8]} under a short, single-GPU budget. The absolute PPL values are not comparable with other tables; only the reversal penalties are used to assess the interaction.

The reversal penalty is nearly twice as large for softmax as for GLA (+112.4 vs. +62.2 PPL). Thus, sensitivity to the window schedule depends

Kernel	Forward	Reversed	Penalty
softmax	185.5	297.9	+112.4
GLA	230.2	292.4	+62.2

Table 7: Short matched-budget kernel-by-schedule comparison. Reversing the schedule hurts softmax nearly twice as much as GLA.

on the perception kernel, with softmax more affected by reversal in this matched comparison.

A.4 Gating mechanisms

All main experiments use a GRU gate; four ablation comparators are reported in §5.3.

GRU gate (main). The gate controls how the perception output $\mathbf{u}$ combines with the previous state $\mathbf{h}_{t-1}$:

$$\mathbf{z} = \sigma(\mathbf{W}_z[\mathbf{h}_{t-1}; \mathbf{u}]) \quad (\text{update}) \quad (14)$$

$$\mathbf{r} = \sigma(\mathbf{W}_r[\mathbf{h}_{t-1}; \mathbf{u}]) \quad (\text{reset}) \quad (15)$$

$$\tilde{\mathbf{h}} = \tanh(\mathbf{W}_h[\mathbf{r} \odot \mathbf{h}_{t-1}; \mathbf{u}]) \quad (\text{candidate}) \quad (16)$$

$$\mathbf{h}_t = (1 - \mathbf{z}) \odot \mathbf{h}_{t-1} + \mathbf{z} \odot \tilde{\mathbf{h}} \quad (\text{output}). \quad (17)$$

Ablation comparators: **sigmoid** ($g \odot u + (1-g) \odot h$), **highway** (gate depends only on h), **residual** ($h+u$), **overwrite** (u).

Gating-mechanism ablation. We replace the GRU gate with simpler combiners while keeping all other architecture and training settings fixed; Table 8 reports the result. The GRU gate is the only combiner that beats the residual baseline by a clear margin on PPL.

Gate	Params	PPL	Clf Avg	QA F1
GRU	30.8M	60.3	91.2	26.3
Residual (+)	26.1M	76.6	89.9	23.6
SigmoidDropout	27.7M	71.7	89.8	23.4

Table 8: Gating ablation at staged $[8, 32, 128] \times T_s=4$. The plain sigmoid combiner ($g \odot u + (1-g) \odot h$) diverges at step 3k due to unbounded state growth and is therefore omitted from the table.

Step-embedding ablation. We replace the learned per-iteration embedding $\mathbf{s}_t$ with either no step embedding or a sinusoidal one, keeping all

other architecture and training settings fixed; Table 9 reports the result. Some form of parametric per-step conditioning is necessary for the iterated body to work, and the sinusoidal choice in particular is a poor fit under deterministic short-horizon T_s .

Step embedding	PPL	Clf Avg	QA F1
Learned (flagship)	60.3	91.2	26.3
None	82.7	89.9	22.8
Sinusoidal	119.5	88.1	19.4

Table 9: Step-embedding ablation.

T_s **sweep.** We retrain Hier-TextNCA from scratch at $T_s \in \{2, 4, 6, 8\}$ with all other architecture and training settings fixed; Table 10 reports the result. PPL is U-shaped, which rules out a monotone “more iterations helps” reading at fixed parameters.

T_s	Total NCA steps	Params	PPL
2	6	30.83M	80.3
4	12	30.84M	60.3
6	18	30.86M	73.9
8	24	30.87M	117.9

Table 10: Effect of iterations per stage T_s at matched compute. Each setting is a single run. PPL is U-shaped, with the best observed value at $T_s=4$.

A.5 Non-hierarchical orchestration variants

For completeness, three non-hierarchical orchestrations used only as ablation comparators (Table 2 in the main paper):

Single-Layer TextNCA. One perception kernel at window w , applied T times.

MultiLayer TextNCA. N distinct sub-layers per iteration (e.g. 3 sub-layers $\times T=4$), same window size, different projection weights per sub-layer.

Stacked NCA. N distinct cells (different weights) chained, each iterating T_{local} times (variants $6C \times T2$, $12C \times T1$, etc.). All three score substantially worse than the hierarchical staged variant; see Table 2.

Model configuration summary.

B Seed Variance

For Hier-TextNCA, SWin-TF-Staged and Transformer-6L, the main tables show the best

Model	Params	Steps	Key design
TextNCA-L1	21.4M	60k	$w=32, T=12$
TextNCA-GLA	21.4M	60k	GLA kernel
TextNCA-Conv	29.0M	60k	Conv kernel
Hier-TextNCA	30.8M	60k	Staged $[8, 32, 128] \times T_s=4$
Hier-TextNCA (wide-first)	30.8M	60k	Staged $[32, 8, 128] \times T_s=4$
SWin-TF-Staged	35.5M	60k	Layer windows $[8, 8, 32, 32, 128, 128]$
SWin-TF-6L	35.5M	60k	Uniform $w=128$, 6 layers
Transformer-6L	35.5M	60k	6 distinct layers, global attn
Transformer-12L	54.5M	60k	12 distinct layers, global attn
UT	21.4M	60k	Shared weights, global attn
Mamba	~ 30 M	30k	Official HuggingFace implementation
ALBERT-style	19.8M	30k	One block shared across all layers

Table 11: Model configurations. All TextNCA variants use $d=512$, $d_{\text{ff}}=2048$, $H=8$ heads, $d_h=64$.

of three seeds. Table 12 reports all three runs and their mean and standard deviation. Other model comparisons are single runs under the same training setup.

Model	Run 1	Run 2	Run 3	Mean $\pm$ sd
Hier-TextNCA	60.3	72.3	76.2	69.6 ± 8.3
SWin-TF-Staged	64.5	66.0	64.7	65.0 ± 0.9
Transformer-6L	52.8	58.0	58.8	56.5 ± 3.3

Table 12: WikiText-103 PPL over three seeds (42, 1337, 2024) at the 60k-step protocol. The main tables show the best result for each model.

Hier-TextNCA has higher seed variation than the non-iterating control (± 8.3 vs. ± 0.9 PPL). The best-run difference between them is therefore not enough to show that iteration improves perplexity. By contrast, the schedule penalties of $+16.7$ to $+70.8$ PPL are two to 8.5 times the flagship standard deviation and support the schedule result. We do not assign the higher seed variation to a specific architectural component.

C Full Downstream Results

Multi-seed classification. The three-task averages are 91.2 for Hier-TextNCA, 90.5 for Transformer-6L, and 90.8 for Transformer-12L, matching Table 4. These small differences do not establish a classification advantage for Hier-TextNCA.

Hier-Hybrid: a global-attention readout block does not close the QA gap. We tested whether appending a single causal global-attention block plus FFN to the Hier-TextNCA stack at the readout would close the QA gap while preserving the iterated weight-shared local-attention body. Hier-Hybrid (34.0M parameters, $+3.2$ M from the global

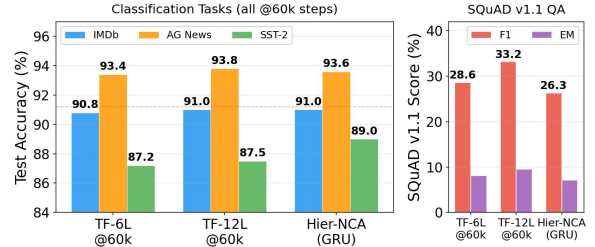


Figure 6: Downstream evaluation at matched 60k training steps. Left: classification accuracy across IMDb, AG News and SST-2; the dashed line marks Hier-TextNCA’s 91.2% three-task average. Right: SQuAD v1.1 F1, where global-attention models reach 27.4–33.2 and local-attention variants reach 19.4–26.3.

block) reaches LM PPL 61.0, classification average 89.9, and SQuAD F1 22.4, compared with 60.3, 91.2, and 26.3 for Hier-TextNCA. This experiment shows that the tested readout modification does not improve QA; it does not establish whether global context elsewhere in the stack would help.

D Training dynamics

Figure 7 plots WikiText-103 PPL against training steps for the key variants in Table 2, referenced from §5.2. Hier-TextNCA GRU and TF-12L are shown as final-step points because their continuous logs were not retained; the remaining curves are full training trajectories clipped to the 60k matched-step budget. SWin-TF-Staged finishes 4.2 PPL behind Hier-TextNCA in the best runs but has the lower three-seed mean (65.0 vs. 69.6); full seed results are in Table 12. The reversed schedule falls early and then plateaus much higher (131.1), so it does not simply need more training. Hier-TextNCA StochasticT trains stably at a higher PPL throughout, the cost of the iteration-count flexibility discussed in §7.

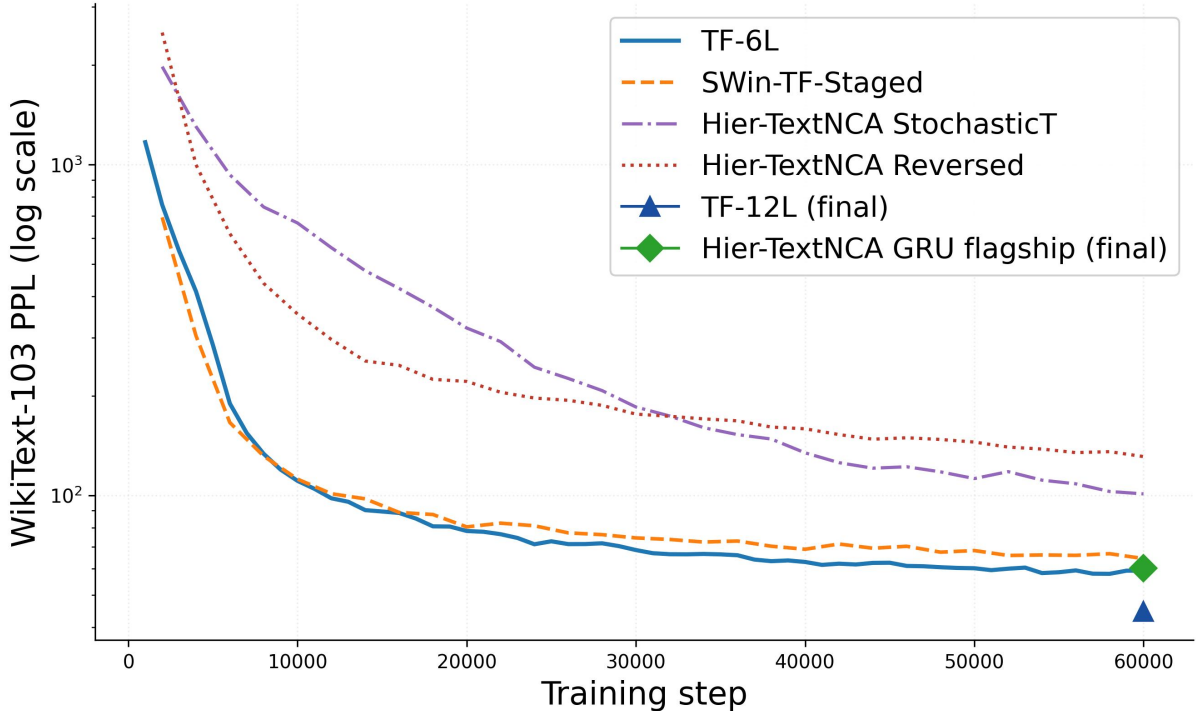


Figure 7: WikiText-103 PPL (log scale) against training steps for the key TextNCA and Transformer variants. Hier-TextNCA GRU flagship (green diamond, PPL 60.3) and TF-12L (blue triangle, PPL 44.7) are final-step-only points; their continuous logs were not retained. TF-6L (blue) uses the full 120k-step log clipped to 60k. SWin-TF-Staged (orange, 64.5) tracks Hier-TextNCA closely, while Hier-TextNCA StochasticT (purple, 101) and Hier-TextNCA Reversed (red, 131.1) illustrate the cost of stochastic- T_s training and of reversing the window schedule.

E Additional Interpretability Analyses

This appendix expands the main-paper interpretability analysis in §6. All analyses use the same Hier-TextNCA checkpoint; the running prompt for per-token figures is “Paris is the capital of France and Berlin is the capital of Germany today.”, chosen because the demonstration pair Paris–France lies 7 tokens before the next-token position, inside $w=32$. The synthetic distance sweep in Appendix F places a 7-token repeat in stage 2, while this prompt requires an analogy completion rather than a literal repeat, so the two measurements are not the same task. We keep this prompt as a qualitative illustration and treat the population sweep as the evidence. The goal is not to assign fixed linguistic labels to stages, but to test whether the staged computation produces distinct internal regimes and whether those regimes explain the per-step loss decomposition discussed in §6.

E.1 Per-step language-modeling loss

Figure 8 reports the per-step next-token cross-entropy referenced from the main paper. We analyse all 12 intermediate NCA states by applying the

shared LM head after each step and measuring next-token cross-entropy on 64 WikiText-103 sequences (32,768 tokens).

The pattern is highly asymmetric. The embedding state $\mathbf{h}_0$ has trivially large loss (39.6 nats) because tied embeddings self-decode toward the current token rather than the next token. The first NCA step collapses this to 9.9 nats, close to the uniform-vocabulary ceiling $\log V \approx 10.4$. Stages 1 and 2 then largely plateau, with per-step changes between -0.3 and $+0.2$ nats. Nearly all meaningful next-token improvement occurs in the final three steps of stage 3: $S3.2 \rightarrow S3.3 \rightarrow S3.4$ reduce loss by $2.4 + 1.9 + 1.5 = 5.8$ nats, taking perplexity on this 32,768-token subset from 18,138 to 59 (within sampling variance of the 60.3 reported on the full WikiText-103 eval set).

E.2 Full representation-similarity matrix

Figure 4 (in the main paper) shows the complete 13×13 linear centred-kernel-alignment matrix over the embedding state and all 12 NCA states, estimated on 2,048 WikiText-103 tokens. The random-pair floor at this sample size is ≈ 0.20 ; all values below should be read against that baseline rather

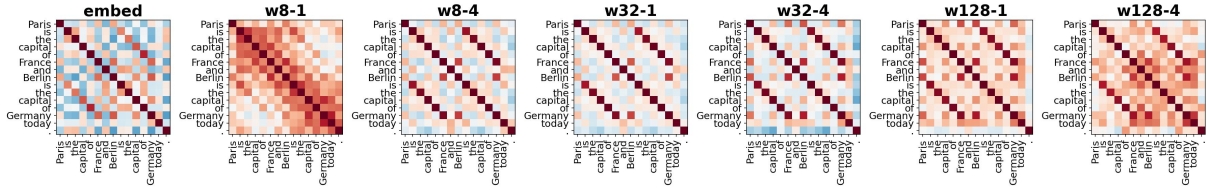


Figure 9: Pairwise cosine similarity between the 15 token hidden states at seven snapshots (embedding + start and end of each of the three stages), colored on $[-0.5, 1]$ (red high, blue low, diagonal = 1). Paris–France $0.48 \rightarrow 0.73 \rightarrow 0.57$, Berlin–Germany $0.71 \rightarrow 0.85$, and the same-syntactic-role pair Paris–Berlin $0.54 \rightarrow 0.75$. The random-pair median also drifts up ($0.22 \rightarrow 0.43$); the two named pairs stay clearly above this baseline at every step.

steps. We describe gate dynamics here, and not a fixed mapping of stages to linguistic categories.

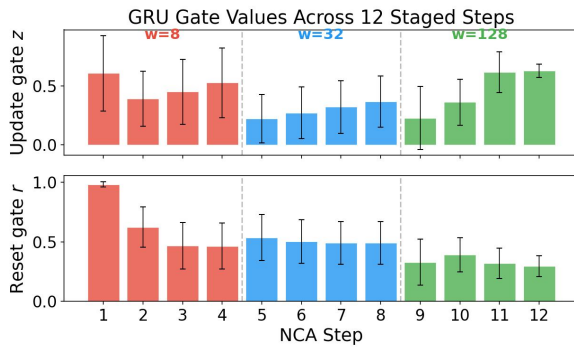


Figure 10: GRU gate statistics per NCA step. Bars are means over hidden dimensions; error bars are $\pm 1\sigma$. Top: update gate z , a small U-shape in the $w=8$ stage ($0.60, 0.39, 0.52$), ~ 0.30 through the $w=32$ stage, and rising to 0.62 by step 12. Bottom: reset gate r , falling monotonically from 0.98 at step 1 to about 0.30 in steps 9–12. The pair encodes an accumulate–consolidate–rewrite role across the three stages.

E.5 Within-stage attention progression

Figure 11 shows head-averaged softmax attention maps within each stage on the running prompt. Each row is a stage (weights shared within a row, windows $w \in \{8, 32, 128\}$); each column is an iteration $1 \rightarrow 4$ within that stage. Despite identical weights, the attention pattern changes across iterations because the underlying hidden states evolve. The cross-clause association is the cleanest case: the attention from “Germany” to “France” is ≤ 0.09 throughout Stages 1–2, where the argmax is the adjacent token “capital”; it then jumps to 0.20 – 0.27 in Stage 3, with “France” itself becoming the argmax. This is an analogy completion rather than a literal repeat, so it is not covered by the window law in Appendix F; we report it as an illustration of how attention changes across stages under identical weights.

E.6 Per-step linear probing

Figure 3 (in the main paper) plots the per-step linear-probe accuracy. A linear probe trained on hidden states at each NCA step (5-way topic classification on a held-out WikiText-103 split) reveals a clear non-monotone profile across stages: accuracy is 56.7% at the embedding, jumps to 77.5% after the $w=8$ stage, peaks at 81.7% during the $w=32$ stage, and stabilizes at 79.2% through $w=128$. Two readings follow. First, the mid-range $w=32$ stage is the most informative for sentence-level topic structure, while the final wide-context stage trades some of that linear separability for next-token decode readiness, consistent with the loss decomposition in §E.1. Second, probing accuracy and LM loss therefore track different aspects of the computation: a stage can be highly probe-accurate (Stage 2) without contributing measurable next-token loss reduction.

E.7 Hidden-state dynamics

Figure 12 tracks the update magnitude $\|\Delta \mathbf{h}\|$ and cosine similarity between consecutive states. Within each stage, cosine similarity increases, indicating local stabilization. At stage boundaries, similarity drops, indicating a shift to a new representational regime. Importantly, the update magnitude does not vanish by the final trained step, so the NCA is not simply converging to a fixed point within the trained horizon.

This helps interpret the iteration ablation: increasing the number of steps is not guaranteed to improve performance by further convergence. Instead, additional untrained iterations can move the hidden state outside the region that the LM head has learned to decode.

E.8 Logit-lens evolution across stages

Figure 13 applies the LM head at each of the 13 representations (embedding $\mathbf{h}_0$ + 12 NCA steps)

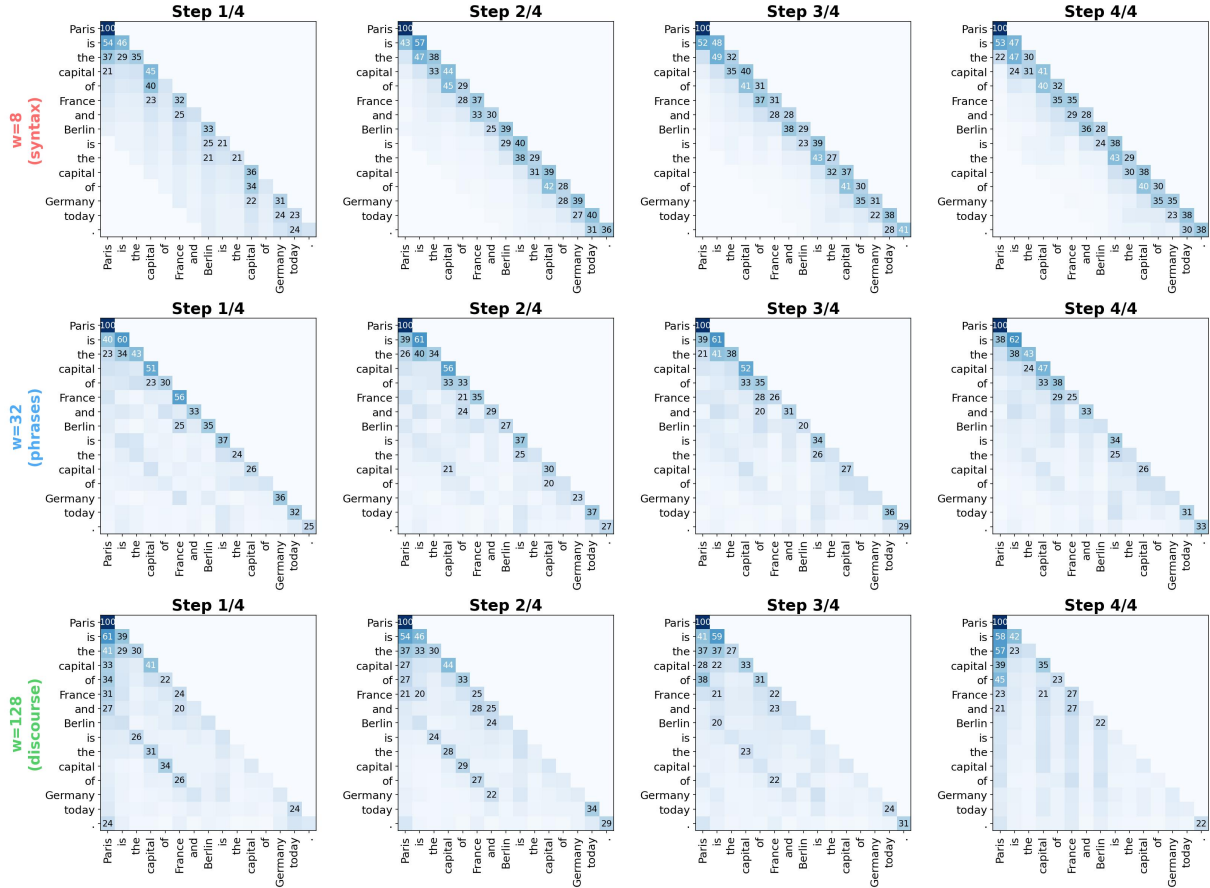


Figure 11: Within-stage attention maps for the running prompt. Rows are stages ($w=8, 32, 128$); columns are iterations 1 → 4. Cell values are head-averaged attention weights shown as percentages (e.g. “27” means 0.27); only cells with weight > 0.20 are annotated. The cross-clause association “Germany” → “France” (7-token distance) is $\leq 9\%$ throughout Stages 1–2 and jumps to 20–27% in Stage 3, becoming the argmax.

for three positions in the science prompt “The theory of relativity states that the speed of light is constant in all reference frames. This implies that time and space are not absolute. . .”. Each horizontal bar shows the top-5 predicted-token distribution at that step; background shading marks stages (red $w=8$, blue $w=32$, green $w=128$). Three positions illustrate the prepare-then-decode pattern. **Position 12** (“constant”→“in”): h_0 self-decodes as “constant” with 88% probability; Stage 1 produces short local continuations (“over”, “or”, “it”); Stage 3 sharply shifts to function-word continuations with “in” reaching the top-5. **Position 17** (“.”→“This”): after the sentence boundary, Stages 1–2 predict punctuation-like continuations (“:”, “-”), while Stage 3 produces sentence-initial tokens “The” / “In” / “This” (7%). **Position 22** (“and”→“space”): only Stage 3 produces content words for the compound continuation, with “space” 21%, “time” 20%, “energy” 8%. Across all three positions, Stages 1–2 rearrange representations and

Stage 3 converts them into usable next-token distributions, consistent with the per-step loss analysis (Figure 8).

E.9 Per-token logit-lens trajectories

Figure 14 traces top-1 logit-lens decodings per token across the 12 NCA iterations on the running Paris–Berlin prompt. At step 0 each position decodes to itself (the embedding). In the $w=8$ stage, tokens drift to punctuation and filler subwords (–, ,, \n); in $w=32$, positions acquire *relational* roles (“which”, “state”, “being”, “borders”) — a syntactic scaffold shared across positions. Only in $w=128$ does content crystallize: the final position settles on “the” (56%) with “France”, “Europe”, “Paris”, and “Germany” in ranks 2–5. This figure should be read as a mechanistic illustration, not a benchmark; the population evidence for the window law is in Appendix F.

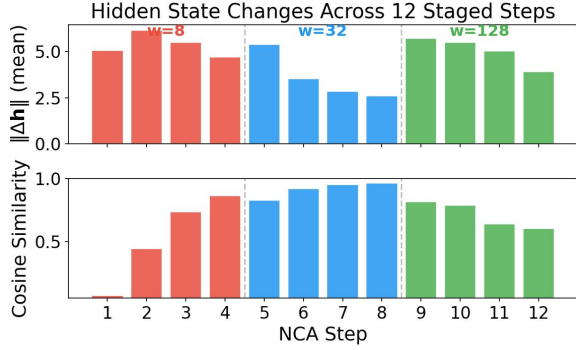


Figure 12: Hidden-state dynamics across the 12 NCA steps. Top: update magnitude $\|\Delta h\|$. Bottom: cosine similarity between consecutive states. States stabilize within stages but shift at stage boundaries; the final update remains nonzero.

E.10 Per-position prediction confidence

Figure 15 gives a per-token view of the prepare-then-decode pattern. Top-1 confidence is high at the embedding state because the tied LM head can decode the input token, but this confidence is not useful for next-token prediction. Confidence then remains low through most of stages 1 and 2, before correct next-token predictions emerge in stage 3 for positions whose continuation is locally well determined.

This complements the aggregate loss curve: early stages destroy the trivial copy prediction and reshape the representation, while stage 3 turns the representation into next-token probability mass.

E.11 Top- k evolution at the final position

Figure 16 zooms into the final position of one illustrative prompt and tracks the top-5 predicted tokens across all snapshots. The correct factual continuation only appears in the top-5 at the final step. This example should be read as an illustration of the receptive-field mechanism, not as a general factual-retrieval result. A stronger claim about factual association would require a probe or benchmark across many prompts.

E.12 Causal stage ablation

We drive the model’s own stage loop and skip individual stages. The full control reproduces the baseline perplexity of 60.5, which validates the procedure. Skipping a stage leaves the hidden state off-distribution for the later stages and for the shared LM head, which is only trained to decode the final stage-3 representation, so the absolute perplexities are not meaningful on their own; they sit just below

the uniform-vocabulary ceiling of about 32,000. Only the ordering of the conditions should be read.

Condition	PPL
Full model	60.5
Skip stage 1 ($w=8$)	9,198
Skip stage 2 ($w=32$)	1,382
Skip stage 3 ($w=128$)	21,160
Stage 3 only, from embeddings	18,122

Table 13: Causally skipping stages. Read the ordering, not the raw values.

Removing stage 3 is the most damaging, which is consistent with stage 3 performing the decoding. Stage 3 on its own also has high perplexity, indicating that the earlier stages are not inert. Skipping stage 1 hurts considerably more than skipping stage 2, so the two preparation stages are not interchangeable. Subject to the off-distribution caveat above, this intervention provides additional support for the prepare-then-decode account.

E.13 Self-repair under corruption

§5.4 reports the headline recovery numbers. We inject a one-shot corruption of controlled magnitude, scaled by the standard deviation of the hidden state, at the first NCA step, and then let the remaining updates run unchanged. Immediately after injection the logit-lens perplexity is roughly 30,000 to 49,000, close to a uniform output distribution. The recovery figures in Table 14 are measured after the final step, against a clean logit-lens baseline of 60.5 PPL on the same evaluation subset. For the Transformer-6L control, we apply the same magnitude-scaled corruption at layer 2 and measure the loss after the rest of its forward pass.

Magnitude ($\times$ state sd)	TextNCA		TF-6L
	Final PPL	Recovery	Final ΔL
0.25	69	98%	+0.09
0.5	114	90%	+0.40
1.0	574	66%	+1.54
2.0	6,317	31%	+3.51

Table 14: Recovery after one-time hidden-state corruption at TextNCA’s first step and Transformer-6L’s second layer. TextNCA’s clean baseline is 60.5 PPL; recovery is the fraction of induced loss removed by the remaining updates. Transformer ΔL is the final loss increase over its clean output, in nats.

We define the metric in terms of three logit-lens cross-entropy values (all measured on the same evaluation subset): the clean final loss L_{clean} (no

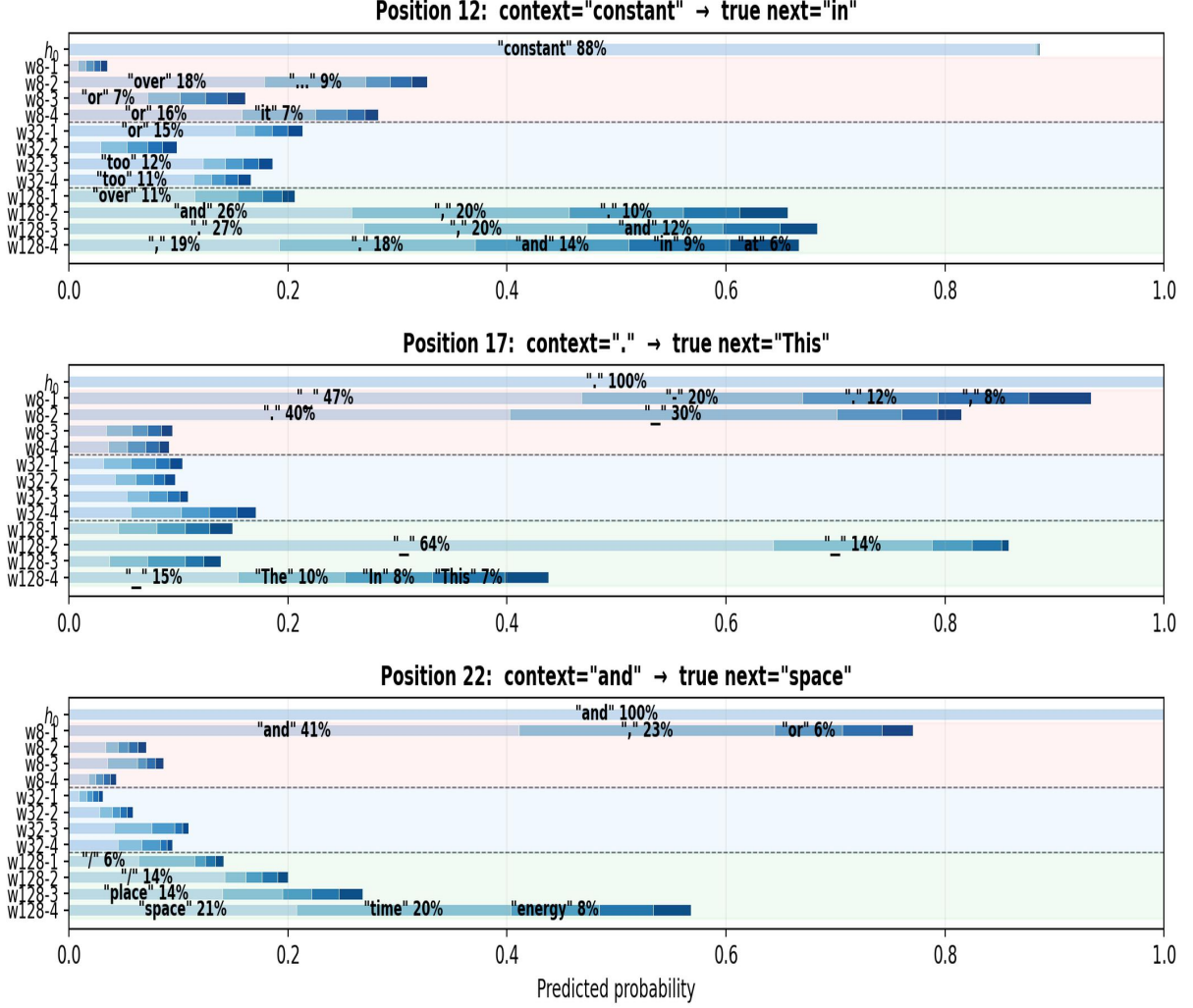


Figure 13: Logit lens across the 13 representations (embedding + 12 NCA steps) for three positions in the relativity prompt. Each bar is the top-5 distribution at that step; shading marks stages (red $w=8$, blue $w=32$, green $w=128$). Position 12: “constant”→“in”. Position 17: “.”→“This”. Position 22: “and”→“space”. Stages 1–2 mostly rearrange representations; Stage 3 produces the usable next-token distribution.

corruption), the peak loss L_{peak} immediately after injection, and the final loss L_{final} after the remaining updates. The *induced loss* is the corruption-induced increase above the clean baseline, $L_{\text{peak}} - L_{\text{clean}}$. The *recovered loss* reported in Table 14 is the fraction of this induced loss that the remaining updates remove,

$$R = \frac{L_{\text{peak}} - L_{\text{final}}}{L_{\text{peak}} - L_{\text{clean}}} = 1 - \frac{L_{\text{final}} - L_{\text{clean}}}{L_{\text{peak}} - L_{\text{clean}}}. \quad (18)$$

$R=1$ means the final loss returns exactly to the clean baseline, while $R=0$ means the corruption persists undiminished to the output. Because the denominator is large (the peak is near a uniform distribution) and the LM head was trained only on final-stage states, a high R mainly reflects climbing back down from a near-uniform output rather than

an exact match to the clean output or reconstruction of the clean hidden state. The experiment covers one corruption family and one injection location in each tested checkpoint, so the numbers should be read as an illustrative output-loss recovery probe.

E.14 Summary

Across CKA (§E.2), semantic-clustering similarity (§E.3), GRU gates (§E.4), within-stage attention (§E.5), linear probes (§E.6), hidden-state dynamics (§E.7), and logit-lens analyses (§E.8–E.11), the stages are clearly not redundant: they produce distinct internal regimes, stage-boundary rewrites, and qualitatively different attention patterns under identical weights. The probe and similarity analyses further show that early stages can be representationally informative (Stage 2 probing peak at 81.7%;

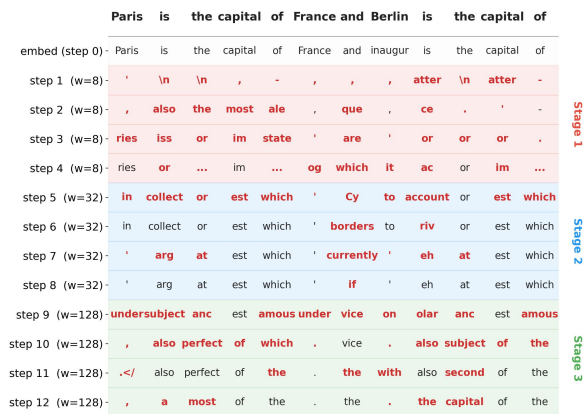


Figure 14: Per-token logit-lens trajectories across 12 NCA iterations on the running prompt. Rows: embedding + 12 steps (stage-shaded). Columns: token positions. Each cell shows the top-1 decoded token from that position’s hidden state; red marks a change from the previous step. Stage 1 produces filler, Stage 2 produces relational connectives, Stage 3 produces content words that match the true completion.

Paris–France similarity rising to 0.73 by the start of Stage 3) without contributing measurable next-token loss reduction. We deliberately avoid overinterpreting these internal changes as independently predictive linguistic stages. Combined with the main per-step loss decomposition, the most conservative interpretation is that Hier-TextNCA performs representation preparation in early stages and next-token decoding in the final wide-context stage.

F Long-Context Modeling and Retrieval

This appendix reports the retrieval measurements referenced from §5.6 and §6. All numbers are inference on frozen 60k-step checkpoints.

F.1 Associative recall on WikiText-103

Whenever a content token reappears, the model must predict the token that followed its previous occurrence. We evaluate 24,875 naturally occurring repeated-token dependencies over 624 sequences, grouped by distance, and report top-5 recall.

Across the tested dependencies up to 127 tokens, the local models remain close to the global ones. Beyond that range, both local models drop sharply while every global-attention model retains higher recall, including the Universal Transformer, which is iterated. The shared drop is therefore consistent with locality, rather than iteration or weight sharing, contributing to the retrieval limit. SWin-TF-Staged,

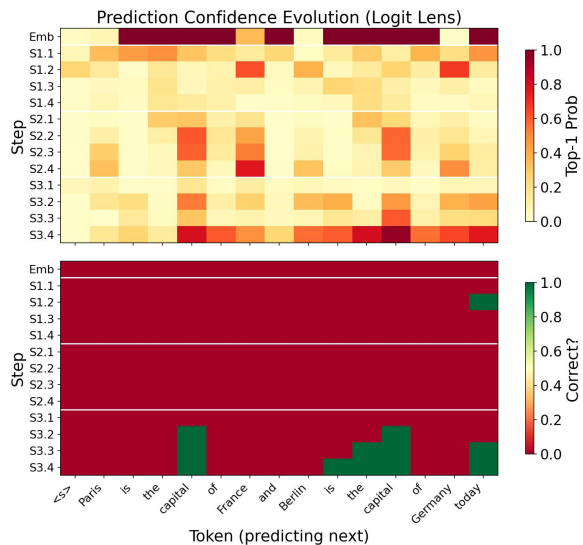


Figure 15: Per-position logit-lens prediction evolution. Top: top-1 probability. Bottom: whether the top-1 prediction matches the true next token. Correct predictions emerge primarily in stage 3.

which does not iterate, also recalls slightly better than Hier-TextNCA in every in-window bucket.

Model	Attn	1–7	8–31	32–127	128–255	256+
Hier-TextNCA	local	0.60	0.70	0.70	0.40	0.40
Transformer-6L	global	0.62	0.72	0.71	0.66	0.63
Transformer-12L	global	0.63	0.71	0.70	0.66	0.63
Universal TF	global	0.64	0.69	0.68	0.61	0.59
SWin-TF-Staged	local	0.67	0.76	0.74	0.39	0.36

Table 15: Top-5 associative recall by key-to-query distance.

Distance d	Forms at	Top-1	Top-5
3	Stage 1	0.72	0.84
5, 7	Stage 2	0.68–0.70	0.80–0.84
8, 10, 14, 20, 28, 31	Stage 2	0.61–0.68	0.74–0.78
32, 40, 56, 80, 110, 127	Stage 3	0.43–0.62	0.61–0.72
135, 160, 200, 260	never	0.00	0.00

Table 16: Synthetic distance sweep, 256 prompts per tested distance. The stage transition occurs between $d=31$ and $d=32$; retrieval succeeds at $d=127$ but not at the next tested distance, $d=135$.

Three findings emerge. First, the association stage follows the active attention window. The transition is sharp: $d=31$ forms in stage 2, whereas $d=32$ requires stage 3. At the upper boundary, retrieval succeeds at $d=127$ but fails at the next tested distance, $d=135$. Second, direct association does not track the additive receptive-field bound created by iteration. Although $d=20$ lies within the stage-1 bound, $R_1 = 1 + T_s(w_1 - 1) = 29$, it

Top-5 predictions at the FINAL position across NCA steps

	rank 1	rank 2	rank 3	rank 4	rank 5
embed -	of 1.00	- 0.00	, 0.00	in 0.00	and 0.00
step 1 (w=8) -	- 0.19	, 0.06	in 0.06	- 0.03	B 0.03
step 2 (w=8) -	- 0.09	- 0.06	M 0.02	love 0.02	F 0.02
step 3 (w=8) -	- 0.09	and 0.05	... 0.03	/ 0.03	a 0.03
step 4 (w=8) -	... 0.04	- 0.03	state 0.02	State 0.02	being 0.02
step 5 (w=32) -	which 0.07	state 0.03	being 0.02	itself 0.02	· 0.01
step 6 (w=32) -	which 0.11	state 0.01	being 0.01	itself 0.01	origin 0.01
step 7 (w=32) -	which 0.09	aters 0.01	ensor 0.01	orbit 0.01	once 0.01
step 8 (w=32) -	which 0.03	aters 0.02	ensor 0.01	isme 0.01	orbit 0.01
step 9 (w=128) -	amous 0.02	», 0.01	ources 0.01	anked 0.01	specialized 0.01
step 10 (w=128) -	the 0.10	which 0.04	a 0.02	its 0.02	his 0.02
step 11 (w=128) -	the 0.10	its 0.03	kingdom 0.01	Arab 0.01	Europe 0.01
step 12 (w=128) -	the 0.56	France 0.09	Europe 0.02	Paris 0.02	Germany 0.01

Figure 16: Top-5 predicted tokens at the final position across the embedding state and 12 NCA steps. The correct continuation appears only at the final step in this illustrative example.

forms only in stage 2. More generally, only $d=3$ forms in stage 1; even $d=5$ and $d=7$ require stage 2. Third, attention association precedes successful decoding. For every reachable distance, B enters the query’s top-five predictions only in the final stage.

F.2 Repeated content words in WikiText-103

We repeat the argmax-hit measurement on natural text. Instead of synthetic pairs we mine every content-token repeat, defined as the nearest previous occurrence of the same token, excluding the 100 most frequent function words, over all 624 test sequences. This gives 24,875 dependencies. At each update, a hit occurs when the query’s head-averaged attention argmax selects either the previous occurrence or the token that follows it. S1–S3 are the hit rates averaged over the four updates in each stage, and “Forms at” is the first stage whose mean reaches 0.15. We also report the final logit-lens top-5 accuracy for predicting the token that followed the previous occurrence.

The same pattern appears on real text: the association forms in stage 2 for distances 1–31, only in stage 3 for distances 32–127, and disappears past the widest window, with the stage-3 argmax-hit rate falling from 0.51 to 0.01 to 0.00. One caveat is

Distance	Pairs	S1	S2	S3	Forms at	acc@5
1–7	959	0.06	0.43	0.62	Stage 2	0.60
8–31	6,601	0.00	0.25	0.54	Stage 2	0.70
32–127	11,236	0.00	0.00	0.51	Stage 3	0.70
128–255	4,222	0.00	0.00	0.01	never	0.40
256+	1,857	0.00	0.00	0.00	never	0.40

Table 17: Attention-argmax association and copy accuracy by distance on WikiText-103. S1–S3 are pair-based argmax-hit rates averaged over the four updates in each stage.

that top-5 accuracy remains near 0.40 beyond 128 tokens because local context can predict some continuations. The pair-based argmax-hit rate, which falls to near zero, is therefore the cleaner retrieval signal.

F.3 Long-form answer modeling on ELI5

We perform an inference-only check on 1,500 examples selected from the sentence-transformers/eli5 training split. We retain examples with answers of at least 120 tokens, concatenate each question and answer, and truncate the sequence to the 512-token context used in pre-training. The resulting sequences have a median length of 191 tokens. For each frozen checkpoint, we compute perplexity over the retained answer tokens, conditioned on the question and preceding answer tokens.

Model	Answer-token PPL
Transformer-12L	40.2
Transformer-6L	45.2
Hier-TextNCA	48.0
SWin-TF-Staged	48.4
Universal Transformer	69.4

Table 18: Answer-token perplexity on 1,500 selected ELI5 training-split examples. Lower is better. Models are evaluated by inference from frozen 60k-step checkpoints without ELI5 fine-tuning.

Hier-TextNCA is 2.8 PPL behind Transformer-6L, close to SWin-TF-Staged, and 7.8 PPL behind Transformer-12L. These results indicate no catastrophic degradation on the selected long-form samples. They do not measure generated-answer quality, constitute a held-out ELI5 benchmark, or establish a long-context advantage. We omit the preliminary effective-context curve because truncating each example to its most recent K tokens changes the learned positional indices, and many examples are shorter than the largest tested values

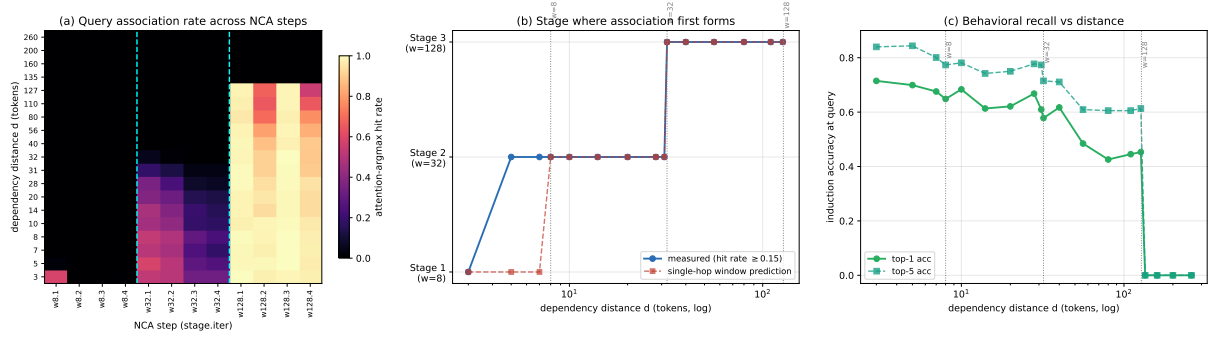


Figure 17: Controlled retrieval as a function of dependency distance (256 synthetic prompts per distance; sequence length fixed at 288). (a) Fraction of prompts for which the query’s head-averaged attention argmax selects either member of the earlier (A, B) pair at each NCA step; cyan lines mark stage boundaries. (b) First stage where this association rate reaches 0.15, compared with the single-hop window prediction. (c) Final top-1 and top-5 recall. The measured stage changes between $d=31$ and $d=32$, and recall falls to zero after $d=127$.

of K ; both effects confound its interpretation.

G Test-Time Iteration-Count Control: Full Sweep

§7 reports the main iteration-count result. This appendix provides the full inference-time T'_s sweep, the deterministic baseline that motivated stochastic-iteration training, and an analysis of why the deterministic model does not generalise across iteration counts.

G.1 Why the deterministic flagship has no inference-time knob

We trained Hier-TextNCA with $T_s = 4$ iterations per stage (12 total NCA steps) and evaluated whether its inference iteration count can change without retraining.

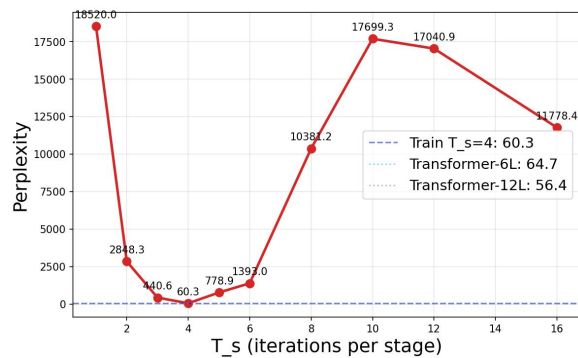


Figure 18: WikiText-103 PPL (log scale) for the deterministic Hier-TextNCA flagship evaluated at different inference T'_s (total NCA steps = $3T'_s$). PPL is U-shaped with the minimum at the training horizon ($T'_s=4, 12$ steps).

Figure 18 shows the result is negative: perplexity is sharply U-shaped, bottoming at $T'_s = 4$ (PPL

60.3) and exceeding 10,000 at $T'_s \in \{1, 8, 10, 12\}$. Two mechanisms explain this.

Learned step embeddings break extrapolation.

Each of the 12 iterations has its own learned additive embedding s_t . Values of t outside $[0, T_s - 1]$ within a stage are never seen during training, so evaluating with $T'_s > 4$ feeds the perception kernel an out-of-distribution conditioning signal.

No fixed point at the trained horizon. As shown in §E.7, $\|\Delta h\| \approx 4$ even at step 12. Iterating beyond 12 compounds this drift rather than converging, so the hidden states leave the region the LM head is calibrated to decode.

G.2 Stochastic- T_s training across iteration counts

Motivated by the deterministic negative result, we train a variant (Hier-StochasticT) with two modifications: (i) sinusoidal step embeddings (deterministic in t , so $T'_s > T_s$ at inference is no longer out-of-distribution conditioning); and (ii) per-batch stochastic sampling $T_s \sim \text{Uniform}\{2, 4, 6\}$ during training, so the shared block is explicitly trained to be useful at multiple iteration counts. All other architecture and hyperparameters are identical to the flagship.

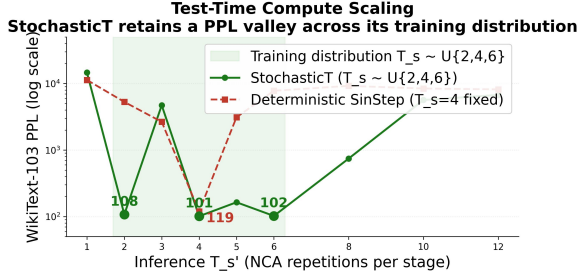


Figure 19: Inference-time T'_s sweep, WikiText-103 PPL (log scale). The shaded band marks the StochasticT training distribution, $T'_s \in \{2, 4, 6\}$. StochasticT (green) remains at PPL 101–108 across these counts; the deterministic SinStep counterpart (red dashed) exceeds 10^3 PPL at every $T'_s \neq 4$.

T'_s	Total NCA steps	StochasticT PPL	SinStep (det.) PPL
1	3	<i>diverged</i>	<i>diverged</i>
2	6	108	<i>diverged</i>
3	9	<i>diverged</i>	<i>diverged</i>
4	12	101	119.5
5	15	164	<i>diverged</i>
6	18	102	<i>diverged</i>
8	24	745	<i>diverged</i>
10	30	<i>diverged</i>	<i>diverged</i>
12	36	<i>diverged</i>	<i>diverged</i>

Table 19: Test-time T'_s sweep on WikiText-103. StochasticT (sinusoidal steps and $T_s \sim \mathcal{U}\{2, 4, 6\}$ training) remains at PPL 101–108 across the trained counts and partially generalises to $T'_s=5$. The deterministic SinStep counterpart exceeds 10^3 PPL at every $T'_s \neq 4$. Bold rows mark counts in the StochasticT training distribution. We report values past PPL 10^3 as *diverged* because exact values in this near-uniform-output regime are not meaningfully comparable (raw values: SinStep $\in [2654, 11310]$; StochasticT off-distribution $\in [4714, 14629]$).

Headline finding. StochasticT remains at PPL 101–108 for $T'_s \in \{2, 4, 6\}$ and reaches 164 PPL at $T'_s=5$. Performance degrades outside this range. The model therefore supports inference across its trained iteration counts, but its best PPL (101) remains 41 points above the deterministic flagship (60.3).

Scope. In our setting, stochastic- T_s training supports inference across the iteration counts used during training. We did not test removing step conditioning or adding an explicit convergence criterion, so these remain open alternatives.

H FLOP- and Wall-Clock-Matched Comparison

The main results in Table 2 report numbers at matched 60k training steps. Because Hier-

TextNCA runs $1.9\times$ more FLOPs per forward pass and is $2.7\times$ slower in throughput than Transformer-6L, matching training steps overstates the case for Hier-TextNCA under a typical compute budget. We re-tabulate here under the two stricter protocols, surfaced in §5.2.

Match	Model	Steps	GFLOPs	PPL
Steps (main)	TF-6L	60k	2,366	52.8
	Hier-TextNCA	60k	4,531	60.3
Train FLOPs	TF-6L	60k	2,366	52.8
	Hier-TextNCA	31k	2,341	70.6
Wall-clock	TF-6L	60k	2,366	52.8
	Hier-TextNCA	23k	1,737	80.1

Table 20: Hier-TextNCA vs. Transformer-6L at three matching protocols. Under both stricter budgets (FLOPs and wall-clock) the gap to TF-6L widens to 17.8–27.3 PPL. PPL numbers for Hier-TextNCA at non-final step counts are interpolated from the eval-loss curve recorded by the trainer.

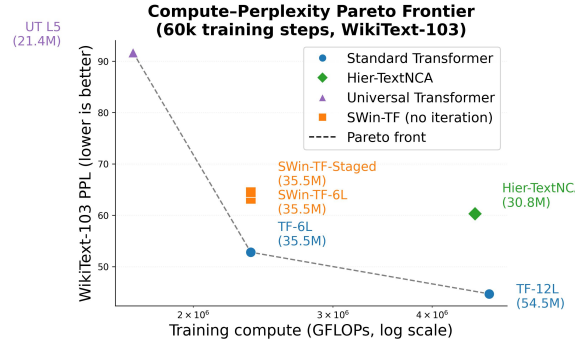


Figure 20: Compute–PPL Pareto frontier on WikiText-103 at 60k training steps. Standard transformers (blue) define the frontier; Hier-TextNCA (green) lies 7.5 PPL behind TF-6L at $1.9\times$ the compute. Dashed line: lower-PPL convex hull. TextNCA is not on the compute–PPL frontier at this scale.

I Reproducibility

We release training code, configuration files (one per row of every results table), and one trained checkpoint per architectural variant. The release documentation will state the license and intended use of each artifact. The training entry point is a single torchrun script with one argument selecting the model from a registry; each row of Table 2, Table 10, and the gating and orchestration ablations corresponds to a registry key documented in the README. Eval scripts produce the JSON final_results.json files that populate every reported number.

Determinism. All runs use a fixed seed of 42 for the trainer; per-rank seeds are 42 plus the rank for data shuffling. We do not enable fully deterministic CUDA kernels because they degrade throughput; rerun-to-rerun variation on a single seed is below 0.05 nats on eval loss in our setup.

Hyperparameters and seeds. All hyperparameters are listed in the per-run config.json: per-GPU batch size, gradient accumulation, world size, effective tokens-per-step, learning rate, warmup, total steps, AMP dtype, and tokenizer. Numbers reported as “mean $\pm$ sd” are over three random seeds; all others are single-seed and are marked as such in the relevant table caption.

Comparison protocol. The 60k main comparison holds data, tokenizer, sequence length, optimizer and training-step budget fixed. The 30k baseline comparison and the FLOP- and wall-clock-matched analyses use separately labelled budgets (Appendix H). We therefore interpret each comparison only within its stated protocol.

Datasets and artifact terms. We use publicly distributed datasets and the Mistral tokenizer for non-commercial academic research, following their respective terms and standard tasks. We collect no new data and apply no filtering beyond upstream dataset processing. Because SlimPajama is web-derived, upstream filtering does not guarantee the absence of personal, offensive, or otherwise harmful content. Dataset sources and statistics are cited in §4; release documentation will record the licenses and intended uses of our code, configurations, and checkpoints.

We do not re-tune each ablation. Every ablation keeps the same optimizer, learning-rate schedule, total steps, batch size, tokenizer, data and seed as the flagship. This is deliberate: re-tuning each variant separately would mix the effect of the component with the effect of its new hyperparameters, so the measured difference could no longer be attributed to the component. As a check on whether the perturbed variants are simply under-optimised, we retrained the largest ablation, schedule reversal, at a second learning rate. The penalty does not close; it grows slightly, from 131.1 PPL at 6×10^{-4} to 139.9 PPL at 1×10^{-3} .

Downstream fine-tuning hyperparameters. For the three classification benchmarks (IMDb, AG News, SST-2), each pre-trained checkpoint

is fine-tuned with AdamW for 5 epochs at learning rate 2×10^{-5} and batch size 32, with a mean-pooled classification head added on top of the LM body and trained jointly. For SQuAD v1.1, we fine-tune for 3 epochs at learning rate 3×10^{-5} and batch size 16 with the standard span-start / span-end prediction head. All fine-tuning runs reuse sequence length 512 and the Mistral tokeniser from LM pre-training. Classification numbers in Table 4 are means over the three seeds (42, 1337, 2024); SQuAD is single-seed.

J Compute Budget

All training runs were performed on a single node with two NVIDIA RTX PRO 6000 Blackwell Workstation GPUs (96 GiB each). Per-run compute is summarised in Table 21; numbers are wall-clock and include checkpoint I/O.

Run	Steps	GPUs	GPU-h
Hier-TextNCA $T_s=4$ (flagship)	60k	2	5.5
Hier-TextNCA $T_s=2$ (ablation)	60k	1	1.4
Hier-TextNCA $T_s=6$ (ablation)	60k	1	4.0
Hier-TextNCA $T_s=8$ (ablation)	60k	1	4.7
Hier-TextNCA gate / kernel abl.	60k	2	5–6
Single-scale TextNCA variants	60k	1–2	2–4
Transformer-6L (baseline)	60k	2	2.4
Transformer-12L (baseline)	60k	2	4.0
Universal Transformer (baseline)	60k	1	~ 1.8

Table 21: Approximate wall-clock GPU-hours per training run; GPU-h is per run for single-design rows and per variant (each) for the two grouped rows (gate / kernel ablations; single-scale variants). The T_s ablations were trained at matched effective batch (65,536 tokens/step) on a single GPU. Total project compute including figures-only and interpretability re-runs is approximately 80 GPU-hours.

Inference cost. A forward pass of Hier-TextNCA at sequence length 512 takes 75.5 GFLOPs versus 39.4 GFLOPs for Transformer-6L and 78.7 GFLOPs for Transformer-12L. End-to-end throughput at batch 32, sequence length 512, on one Blackwell-6000 is 53k tokens/s for Hier-TextNCA versus 138k tokens/s for Transformer-6L ($2.6\times$ slower). All FLOP counts use standard self-attention plus FFN accounting and are reproducible from the released configs.

K Use of AI Assistants

We used AI coding assistants during this project for two purposes: helping write and debug implementation code (model definitions, training and

evaluation scripts, figure-generation scripts) and assisting with paper writing (LaTeX formatting, editorial revisions, consistency checks across sections, and table formatting). The ideation, research questions, experimental design, choice of ablations, the architectural design of TEXTNCA, and the interpretation of all empirical results are the authors' own. All AI suggestions were reviewed and verified by the authors before being committed to the codebase or to the paper.