STRUCTURE BEFORE ATTENTION: A ZERO-INITIALIZED CONVOLUTION RECOVERS WINDOWED LANGUAGE-MODEL QUALITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Sliding-window attention reduces long-context language-modeling cost but can worsen perplexity. We study whether a small convolution added to the residual stream before attention can offset this increase, and what contributes to its benefit. Under paired-seed attribution, paired runs share initialization, data order, and evaluation batches, differing only by the initially inactive branch. Under dense attention the branch gives modest gains at 81M and 253M parameters. In 81M models on PG-19 book text with an 8192-token context and a 256-token window in eight of ten layers, it lowers perplexity by 2.1%, against 0.6% under dense attention, and the windowed model with the branch finishes within 0.16% of the dense model with it in each of three seeds. Across contexts and window sizes, the extra gain descriptively tracks the window penalty (the baseline cost of using sliding-window attention). A pointwise (width-one) branch, which mixes no neighboring tokens, recovers most of the gain, and in a single-seed test, removing the branch after 60% of training and training on retains most of its advantage. The intuition that dense attention learns the branch's function while windowed attention cannot motivated an amplitude-sensitivity test whose registered predictions fail; the mechanism behind the amplified benefit remains open.

1 INTRODUCTION

Language models increasingly serve as agents that read tool outputs and carry long interaction histories, making long-context attention a serving bottleneck. Dense attention, where each token attends to every earlier token, costs time quadratic in context length, so a growing number of production models window most layers or interleave recurrent layers (Jiang et al., 2023; Gemma Team, 2024; Lieber et al., 2024; NVIDIA, 2025), sometimes at a cost in perplexity (Beltagy et al., 2020; De et al., 2024; Ren et al., 2025). A small convolution before attention offers a simple possibility: supply local structure cheaply enough to recover some of that loss (So et al., 2021; Allen-Zhu, 2025; Tian et al., 2026). Yet under dense attention these gains are small, smaller in our runs than the variation between seeds, and how they change with the attention window and the training context remains unclear. We study these conditions with paired-seed attribution, a protocol of paired training runs, asking when the addition helps and whether its benefit requires mixing neighboring tokens at all.

The proposed benefit is that a cheap local branch hands attention a representation in which each token already carries its immediate neighbors, so attention need not spend capacity assembling local structure. If that is what it does, its value should be largest where attention can assemble the least: under a window, and at contexts long enough for the window to bind. The candidate, adapted from the Canon-A layer of Allen-Zhu (2025), is a depthwise causal convolution added to the residual stream before attention, whose taps (the per-channel filter coefficients, one per lag) start at zero; § 4 gives its placement, how it differs from Canon-A, and its cost. But the evidence is weakest where this intuition predicts the most. On restricted mixers such as linear attention the reported gains are large; on the standard dense-attention transformer the improvements reported by Allen-Zhu (2025) are of the same order as the seed variation we measure (§ 5.1); and the one controlled comparison under windowed attention we know of (Ren et al., 2025) uses a single training context and a different placement.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

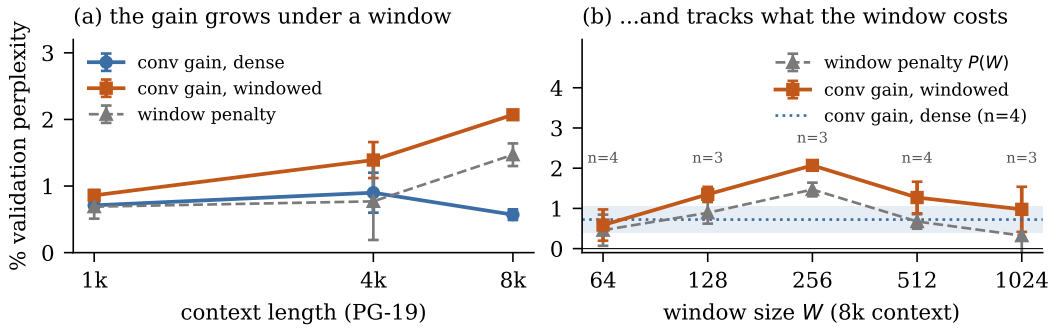


Figure 1: Extra convolution gain tracks window penalty descriptively on PG-19: (a) across contexts; (b) across window sizes at 8192 tokens. W: window size in tokens. Points: means; bars: sample standard deviation (SD). n=3 at W=128, 256, and 1024; n=4 at W=64 and 512. In (b), the dotted line and band give the dense convolution gain, mean $\pm$ SD over the n=5 rescored dense pairs of the pooled stacks (Tables 8, 9, and 7). All but W=256 pool two training stacks (hardware and precision; Table 9). The plot's $P(W)$ denotes $P_{\text{base}}(W)$.

Two gaps therefore remain: resolving the small dense-attention effect, and determining how it changes under windowing. We address measurement first, because the comparisons across conditions depend on it. Surveys of empirical practice find that variation from seeds and data order routinely exceeds the improvements papers report (Bouthillier et al., 2021), and our own runs bear this out: two identically configured 81M models trained from different seeds finish 2.2% apart in training-time validation perplexity, which also samples different validation text per seed, and even on one fixed validation set, PG-19 baselines from different seeds differ by up to 0.4%, comparable to the dense-attention effect. An unpaired comparison cannot resolve such effects without many replicates.

The idea behind paired-seed attribution is to make the candidate module the *only* difference between two runs. The branch's output is zero at initialization, so the augmented model computes exactly the baseline's function, and its parameters come from dedicated generators that leave the baseline's random stream untouched, so shared weights, data order, and evaluation batches are bitwise identical between the paired runs. The runs separate only through the branch's learned contribution. Paired-seed attribution removes differences in initialization and data order within a pair; replication across seeds measures seed-by-treatment variation, and execution nondeterminism remains (0.1–0.3% between duplicate runs, § 7).

We apply paired-seed attribution to models with 81M–253M parameters trained on 1.6–5.2B tokens to determine when the convolution helps (Figure 1). Under dense attention, it reduces perplexity by 0.5–1.0%, replicated across seeds and contexts. With an 8192-token context and a 256-token window, it reduces perplexity by 2.08 ± 0.04%, a gain $3.5\times$ as large, and the windowed model with the convolution finishes within 0.16% of the dense model with it in every seed; these windowed results are at 81M. Across corpus–context–seed combinations and window sizes, the extra gain under windowing descriptively tracks the baseline cost of windowing.

The practical benefit is clearer than its explanation. A width-one branch, which cannot mix across token positions, recovers 80–86% of the windowed gain, so mixing each token with its neighbors is not required for most of the gain, although it adds a smaller increment that replicates across seeds. In single-seed mid-training controls, a windowed model whose branch is removed after 60% of training and trained on for the remaining 40% keeps 74% of its advantage, while a branch inserted late into a dense model recovers about a fifth of its gain. Under dense attention, the gain is concentrated on novel-bigram tokens, whose preceding-token–target pair has not appeared earlier in context, and is matched by other local operators, none of which shares the convolution's windowed amplification. The extra windowed gain has similar point estimates across bigram-recurrence classes; per token it is largest on tokens that need long context, but most of it falls on the far more numerous tokens that need little. An amplitude-sensitivity test of attention substitution fails its registered predictions (Appendix E); the interaction's mechanism remains open.

Our contributions establish the protocol, its measured effects, and controls on their interpretation:

1. Paired-seed attribution, a paired training protocol for resolving small architectural effects (§ 3).[1]
2. Replicated measurements under dense and windowed attention (§§ 5.1–5.2).
3. A study of how the extra gain changes with context, window size, and training budget (§ 5.3).
4. Controls probing temporal mixing, initialization, and training trajectory (§ 5.4, Appendix E).

2 Related Work

Convolutions before and around attention. So et al. (2021) found depthwise convolutions on Q/K/V through architecture search; short causal convolutions are also part of H3 (Fu et al., 2023) and Hyena (Poli et al., 2023). Canon layers (Allen-Zhu, 2025) systematized residual causal convolutions, with larger gains under linear than dense attention; their Canon-A convolves the normalized attention input, whereas ours adds to the residual stream (§ 4). Ren et al. (2025) report a 2.6% gain from short convolutions on Q/K/V of a windowed model at one training context; placement sweeps (Tian et al., 2026) and dynamic convolutions (Sieberling et al., 2026; Wu et al., 2019) extend the design space, and Sieberling et al. (2026) report a width-one control that keeps most of the gain. Appendix C compares these quantitatively. We hold the module fixed and vary the *conditions* instead.

Hybrid attention and recurrent architectures. Windowed-attention layers with periodic dense-attention layers are standard in efficient language models (Beltagy et al., 2020; Jiang et al., 2023; Gemma Team, 2024; De et al., 2024; Ren et al., 2025), and Cabannes et al. (2025) study how the window size trades short- against long-context ability. Recent hybrids instead interleave a few dense-attention layers with state-space or linear-recurrent layers (Gu & Dao, 2023; Dao & Gu, 2024; Lieber et al., 2024; Yang et al., 2025; Dong et al., 2025; NVIDIA, 2025); those recurrent blocks carry a width-four depthwise causal convolution of their own, the same operator we study, while their attention layers do not. The only windowed ablation we know of (Ren et al., 2025) covers one context; a factorial under paired-seed attribution of ($\pm$ convolution) $\times$ (dense $|$ windowed) $\times$ training context is new. Qiao et al. (2026) argue that hybrid gaps shrink with training budget, a risk we test directly.

Why local mixing should matter. Arora et al. (2023) attribute most of the convolution–attention gap to repeated-bigram "associative recall" tokens. These use an earlier token pair to predict its second token when the first recurs. Induction-head analyses (Olsson et al., 2022; Voita et al., 2019) show early layers assembling previous-token structure that later attention consumes. These motivate our bilinear bigram candidate (§ 5.4) and the prediction that restricting attention raises the value of external local structure; § 5.5 qualifies the first account.

Identity-at-initialization insertion. Function-preserving insertion goes back to Net2Net (Chen et al., 2016) and zero-initialized residual branches are standard for stability (Bachlechner et al., 2021; Zhang et al., 2023); we use them as an attribution device with random-stream discipline.

3 The Paired-seed attribution Protocol

Paired-seed attribution removes initialization, data-order, and evaluation-batch variation between runs, leaving the tested module and floating-point nondeterminism. Let $B(\theta)$ be a baseline transformer with parameters θ drawn from seed s. An admissible candidate has the residual form $x \mapsto x + \alpha \odot f_\phi(\mathrm{RMSNorm}(x))$, with a learned per-channel gate α. Initially, $\alpha = 0$, so the augmented mapping equals the baseline mapping. The parameters ϕ must come from a dedicated generator that never advances the global random stream. The plain convolution preserves the initial mapping through zero taps rather than a zero gate. The structured model $S(\theta, \phi)$ satisfies:

[1] Code: `https://anonymous.4open.science/r/Structure-Before-Attention-E24C`.

- **P1 (identity at initialization):** S's forward pass equals B's bitwise at step 0.
- **P2 (shared randomness):** θ, the data-loader order, and the evaluation batches are bitwise identical between the paired runs.
- **P3 (attribution):** given P2, the divergence of the paired validation curves estimates the total effect of adding and training the branch (including its effect on the shared weights' optimization), up to floating-point nondeterminism.

P1–P2 are verified by exact-equality tests on all shared state and on step-zero logits. We report paired relative perplexity reductions $\Delta = (\mathrm{ppl}_B - \mathrm{ppl}_S)/\mathrm{ppl}_B$ at matched steps, where ppl denotes perplexity. We also report absolute per-token negative log-likelihood (NLL) reduction, Δ_{NLL}, in nats per token (natural-log units, so $\mathrm{ppl} = e^{\mathrm{NLL}}$ and a reduction of 0.01 nats is about a 1% perplexity reduction), and replicate pairs across seeds to estimate the effect's spread.

Paired-seed attribution improves resolution without extra runs (Bouthillier et al., 2021). Training-time baseline perplexities differ across seeds by 2.2–4.9% in our grids; much of this is evaluation-sample variation, since each seed draws different validation windows, and on one fixed validation tiling PG-19 baselines from different seeds still differ by 0.2–0.4%. Figure 2 (Appendix A) shows both runs of each pair: within a seed the curves track each other and the gain stays positive, while comparing runs across seeds gives gains of either sign. Duplicate runs differ by 0.09–0.32% in five cases, a repeatability diagnostic rather than a calibrated floor (§ 7). The estimand is the expected paired delta over training seeds for a fixed architecture, budget, training stack, and validation set; each seed contributes one paired delta. We report mean ± sample standard deviation (SD) across seeds and nominal two-sided 95% t-intervals, which at n=3 assume approximately normal seed effects. Rescored contrasts share one fixed validation tiling, so their intervals cover training randomness, not the sampling of books. Predictions were recorded in the project log, in several cases after the test runs had launched but before they completed; Appendix G gives dates, what was already observed, the original decision rules, and outcomes under them.

4 Experimental Setup

Using paired comparisons, we establish the dense-attention effect, then vary branch presence, attention span, and training context; controls on the branch and its training history help interpret their interaction. All comparisons match training-token budgets across the paired model configurations. OpenWebText (Gokaslan & Cohen, 2019) supports the dense-attention comparisons, while PG-19 (Rae et al., 2020) provides longer documents for the context-length study. Paired models use identical validation batches; selected checkpoints are also deterministically rescored on fixed validation windows with the same procedure. Table 1 summarizes the design and implementation.

The candidate is a width-four depthwise causal convolution with zero-initialized taps (per-channel filter coefficients, one per lag), inserted on the residual stream before attention in every block: $u = x + \mathrm{Conv}_4(\mathrm{RMSNorm}_\gamma(x))$, followed by $y = u + \mathrm{Attn}(\mathrm{RMSNorm}(u))$; γ denotes learned per-channel RMSNorm gains. The branch therefore feeds both the attention input and the residual bypass. Canon-A (Allen-Zhu, 2025) instead computes $y = x + \mathrm{Attn}(h + \mathrm{Conv}_4(h))$ with $h = \mathrm{RMSNorm}(x)$ and Kaiming-initialized taps, leaving the residual stream unchanged; our results need not transfer to that placement. It adds 3,200 parameters per block and increases forward multiply–adds by 0.03% at hidden width d=640, distinct from measured training time: about +15% per step at 1024–2048-token contexts and +12–13% at 8192 tokens (Table 14); at 4096 tokens, measured overhead is within variation between nodes.

5 Results

5.1 Dense attention retains a modest gain

We first test for a gain under dense attention once runs are paired, establishing the reference for window-specific amplification. The pre-attention convolution reduces perplexity by 0.5–1.0% under dense attention at a matched token budget, with replication across seeds and contexts (Table 2). The model with convolution leads at every matched evaluation point of every pair (25 per 81M pair, 20 for the 253M pair; verified programmatically). At 253M parameters and 5.24B tokens, a 3× step

Table 1: Experimental setup; budgets are matched within each paired comparison. Table labels use "+conv" for the added convolution; in windowed models, every fourth layer uses dense attention. Training-step counts use k for thousands.

Component	Setting
Models	Llama-style decoders: RMSNorm, RoPE, flash attention, SwiGLU, tied embeddings; 81M (d=640, 10 layers) and 253M (d=1024, 16 layers).
Attention	Dense: full causal attention over the whole context in every layer. Windowed: each token attends to itself and the previous $W-1$ tokens (W=256), with dense attention in every fourth layer; compiled FlexAttention block masks. Section 5.3 varies W.
Module	Width-four depthwise causal convolution before attention in every block; 2,560 taps and 640 gains γ, totaling 3,200 parameters per block at d=640.
Corpora	OpenWebText: GPT-2 BPE, 9.03B tokens. PG-19: 3.06B tokens, 58 whole books held out, contexts 1024/4096/8192 tokens.
Budgets	81M: $25\text{k} \times 65{,}536 = 1.638$B tokens ($\sim$20 tokens/parameter); 253M: $20\text{k} \times 262{,}144 = 5.24$B tokens.
Optimization	AdamW; cosine $3{\cdot}10^{-4} \to 3{\cdot}10^{-5}$; bf16, DDP on 2–4 A100-80GB. V100-marked runs use fp16; V100 rescoring agrees with A100 within 0.003 perplexity.
Validation	50 shared batches every 1k steps. The grids in § 5.3 and § 5.4 also use 695 nonoverlapping 8192-token windows: all but the last 6,975 of 5.70M PG-19 validation tokens. Rescoring shifts levels by -0.7 to $+0.4$ perplexity, depending on context and checkpoint, and moves each PG-19 contrast by at most 0.21 pp (percentage points).

Table 2: Final perplexities and paired relative reductions under dense attention at matched token budgets (81M, 1.638B tokens on OpenWebText; one 253M pair at 5.24B tokens). Baseline variation across seeds is 2.2%. Three-pair gains: $+0.86 \pm 0.32\%$ at 1024 and $+0.85 \pm 0.11\%$ at 4096. Two-sided 95% t-intervals (mean ± 2.48 SD): $[0.07, 1.64]\%$ and $[0.57, 1.13]\%$, respectively.

pair	Context (tokens)	baseline perplexity	convolution perplexity	Δ (%)
seed 42	1024	28.41	28.12	+1.04%
seed 43	1024	27.78	27.64	+0.49%
seed 44	1024	28.29	28.00	+1.04%
seed 42	4096	27.10	26.90	+0.72%
seed 43	4096	27.50	27.24	+0.92%
seed 44	4096	27.51	27.26	+0.90%
seed 42, 253M	2048	18.45	18.26	+1.05%

in scale and budget, the convolution reduces perplexity by 1.05%; the single pair shows persistence rather than established scale invariance.

Each pair's gain peaks early (4.8% at 2k steps), then declines toward 1%. This is consistent with, but does not establish, attention gradually learning the local structure supplied by the convolution. These comparisons establish a modest dense-attention benefit. We next ask whether restricting attention adds a further benefit.

5.2 Windowing amplifies the gain at long context

For every corpus, context length, and seed we train a baseline and a +conv model, each with dense and with windowed attention. Paired comparisons give the gains Δ(dense) and Δ(windowed). Their difference, the interaction I_{ppl} (pp), is how much more the convolution helps a windowed model; the window penalty P_{base} (%) is how much worse the windowed baseline is than the dense one (Table 3). In NLL, $I_{\text{NLL}} = P_{\text{base,NLL}} - P_{\text{conv,NLL}}$ exactly; percentages approximate it.

On web text, windowing imposes little penalty and the interaction is correspondingly small. OpenWebText documents average $\sim$1.1k tokens, and a 256-token window costs at most 0.6%. The interaction replicates at 4096 but not at 1024.

Table 3: Convolution gain under dense and windowed attention, mean ± SD over seeds 42/43/44. I_{ppl}: interaction (pp); P_{base}: baseline penalty (%); I/P: mean per-seed ratio $I_{\text{ppl}}/P_{\text{base}}$, which approximates the share of the windowed baseline's perplexity cost that the convolution recovers (the exact share, $(P_{\text{base}} - P_{\text{conv}})/P_{\text{base}}$, is 104% at 8192 and 22% at PG-19 1024); n/a where the per-seed penalty is near zero or changes sign. PG-19 rows are rescored deterministically on non-overlapping tilings of the whole validation set (Table 1; at 8192, seed-42 windowed pair: the rerun); OpenWebText rows use training-time validation at full precision. Per-seed values: Appendix B. PG-19 interaction 95% t-intervals: $[1.36, 1.60]$ pp at 8192; $[-0.09, 0.30]$ pp at 1024 (includes zero). I/P per seed: 96–110% at 8192, 5–46% at PG-19 1024. PG-19 penalties at 4096 range from 0.15 to 1.27% across seeds; OpenWebText penalties change sign at 4096 and lie between 0.2 and 0.6% at 1024. At 8192, residual penalty $P_{\text{conv}} = -0.06 \pm 0.10\%$ (95% t-interval $[-0.30, +0.18]\%$): relative perplexity of windowed versus dense models with convolution; negative favors windowed.

corpus	Context (tokens)	Δ(dense)	Δ(windowed)	I_{ppl} (pp)	P_{base} (%)	I/P (%)
OpenWebText	1024	$+0.86 \pm 0.32\%$	$+1.06 \pm 0.12\%$	$+0.21 \pm 0.36$ pp	$+0.37 \pm 0.21\%$	n/a
OpenWebText	4096	$+0.85 \pm 0.11\%$	$+1.08 \pm 0.14\%$	$+0.24 \pm 0.08$ pp	$+0.05 \pm 0.38\%$	n/a
PG-19	1024	$+0.76 \pm 0.04\%$	$+0.87 \pm 0.10\%$	$+0.11 \pm 0.08$ pp	$+0.54 \pm 0.17\%$	22.1%
PG-19	4096	$+0.91 \pm 0.30\%$	$+1.43 \pm 0.25\%$	$+0.52 \pm 0.37$ pp	$+0.79 \pm 0.57\%$	n/a
PG-19	8192	$+0.59 \pm 0.09\%$	$\mathbf{+2.08 \pm 0.04}\%$	$\mathbf{+1.48 \pm 0.05}$ pp	$+1.46 \pm 0.14\%$	$\mathbf{102.3}\%$

On book text, the interaction increases between the shortest and longest contexts by far more than its spread; the intermediate estimate does not establish monotonicity. At 8192, the convolution reduces perplexity by $0.59 \pm 0.09\%$ under dense attention and $2.08 \pm 0.04\%$ under windowing, an amplification of about $3.5\times$. With the convolution, the windowed model differs from the dense model by at most 0.16% in every seed; the residual penalty is $P_{\text{conv}} = -0.06 \pm 0.10\%$ (95% t-interval $[-0.30, +0.18]\%$) against a baseline penalty of 1.46%, so the convolution recovers the window penalty up to that interval. At shorter contexts it does not. At 1024 tokens the interaction averages about a fifth of the penalty, though the per-seed ratio ranges from 5 to 46%, and at 4096 the per-seed penalties are too variable to say. The preregistered one-for-one prediction, made for single runs at 1024 and 4096, missed at both (Appendix G).

Across all fifteen corpus–context–seed combinations, the interaction descriptively tracks the window penalty (r=0.84, intercept -0.02 pp). Partly this is arithmetic, since $I_{\text{NLL}} = P_{\text{base,NLL}} - P_{\text{conv,NLL}}$ and P_{conv} varies less than P_{base} (SD 0.32 versus 0.57 pp); the combinations share seeds and setups, so we report r without a significance test.

5.3 The amplification tracks the window's cost

Changing the window size at a fixed 8192-token context changes how much windowing costs, and the convolution's windowed gain follows that cost (Figure 1b). Pooling seeds and training stacks (n=3–4 per window), the window penalty and the windowed gain both peak at W=256 (penalty 1.46%, gain 2.08%) and are smaller at narrower and wider windows (penalty 0.32–0.89%, gain 0.58–1.35%), while the dense gain stays near 0.7%; across the five windows the two means correlate at r=0.92. The pooled means mix window size with the composition of training stacks, whose dense gains differ by up to 0.58 pp, so we treat that correlation as descriptive. Within each stack and seed, every trajectory measured at three or more windows peaks at W=256 in both interaction and penalty (Figure 3), although the interaction's size varies across seeds and stacks. Table 8 gives the seed-42 sweep and Tables 9, 10, and 11 the replications.

Training budget offers another way to change the penalty. In the tested seed, longer training reduced both penalty and interaction. Motivated by Qiao et al. (2026), we increased the budget to $1.8\times$ at W=256; the dense convolution gain grew while the interaction fell by 39%. The windowed model with convolution again achieves slightly lower perplexity than the dense model with convolution (Table 12); one seed at two budgets (cosine schedule stretched to 45k steps) shows a positive interaction remaining, not its eventual fate.

A restricted mixer that pays no penalty gives the complementary case: in an attention–Mamba-2 hybrid (Dao & Gu, 2024) that is $1.9 \pm 0.4\%$ better than dense attention, the branch at its two attention sites gains only $0.16 \pm 0.02\%$ (interaction $+0.09 \pm 0.04$ pp; Table 16).

Table 4: Pointwise and width-four controls at 8192 tokens, $W{=}256$: rescored perplexity, pointwise share, and remaining temporal gain across three seeds and two stacks, plus the seed-43 dense reference. A duplicate convolution run agreed within 0.23% (§ 7).

seed	stack	base	width 4	width 1	pointwise share	temporal (pp)
43	V100	19.146	18.817 (+1.72%)	18.869 (+1.45%)	84%	+0.28 pp
42	A100	19.295	18.909 (+2.00%)	18.987 (+1.59%)	80%	+0.40 pp
44	V100	19.246	18.882 (+1.89%)	18.933 (+1.63%)	86%	+0.26 pp
42	V100	19.285	18.873 (+2.14%)	18.939 (+1.80%)	84%	+0.34 pp
44	A100	19.245	18.839 (+2.11%)	18.918 (+1.70%)	81%	+0.41 pp
43, dense	V100	18.982	18.753 (+1.20%)	18.799 (+0.96%)	80%	+0.24 pp

5.4 Controls on temporal mixing, initialization, and training trajectory

We further ask whether the branch needs to mix neighboring tokens at all. We replace its width-four convolution with a pointwise (width-one) branch on the same zero-initialized residual path, at 8192 tokens and $W{=}256$. The pointwise branch recovers 80–86% of the windowed gain across three seeds (Table 4). The extra reduction from temporal mixing is positive in all five comparisons (0.26–0.41 pp): 0.29 ± 0.04 pp across the three V100 seeds (95% t-interval $[0.19, 0.40]$ pp) and 0.40 pp–0.41 pp in both A100 seeds. The dense pointwise control also recovers most of the gain. Most of the benefit is therefore achievable without mixing neighbors, and temporal mixing adds a smaller, replicated increment. This rules out temporal mixing as a requirement, but it does not show that the width-four branch obtains its gain the same way: the pointwise branch computes $x_{t,c}\,(1 + w_c\gamma_c/\mathrm{RMS}(x_t))$, an input-dependent channel rescaling that changes residual magnitude, channel balance, and downstream normalization.

A related question is whether the gain is specific to the convolution's parameterization. Under dense attention, other operators match it: at 1024 tokens a bilinear operator, a dynamic-tap convolution, and the convolution give 1.03–1.15% (Table 17), and at 8192 tokens the bilinear and dynamic-tap operators match the convolution's dense gain within seed spread. Under windowed attention they do not: across three seeds, the bilinear, dynamic-tap, and sigmoid-gated operators keep 31, 65, and 24% of the convolution's windowed gain, with interactions of -0.17 pp, $+0.27$ pp, and $+0.14$ pp against $+1.17$ pp (Table 18). The amplification is thus not shared by every zero-initialized residual branch: the pointwise linear branch keeps most of it, while these gated operators keep less. Appendix B describes each operator.

If temporal mixing is not required for most of the gain, the next question is whether the added path must remain active. In single-seed mid-training switches, we switch at step 15k of 25k (after 60% of training), preserve optimizer state, and control for restarting with same-step shams (branch unchanged) on the same training stack (Table 13). Removing the windowed branch and training the remaining 10k steps retains 74% of the advantage; inserting a fresh branch into the dense baseline late recovers only 21–23% of its gain. Once acquired, most of the advantage thus survives the branch's removal, carried by the shared weights together with their optimizer state. These runs do not show when the advantage is acquired, and removal and insertion use different attention types, so they are not a symmetric test.

We next ask whether the trained taps themselves carry the advantage: we copy the taps a completed run learned into a fresh run at step 0 (a transplant). In the same single-seed setting, transplanted taps perform worse than zero initialization, although a transplant also changes the branch's initial function, its amplitude, and the training trajectory. Taps from a completed windowed model of the same configuration are 0.10 perplexity worse on the registered training-time metric, at the preregistered materiality margin (gaps of at most 0.06 count as equivalent to zero initialization, gaps of at least 0.10 as material), and 0.083 worse under deterministic rescoring, between the margins; frozen taps from a dense model exceed the materiality margin.

To distinguish exact identity from a small initial branch, we vary the scale of random taps; in this single-seed dense test the branch must start small but need not start at exactly zero. Default Kaiming taps incur an immediate penalty that is never repaid, whereas random taps a tenth the size match zero initialization at every logged step in the single-seed test (Table 17).

Table 5: PG-19 paired NLL reductions, mean ± SD across matched seeds; windowed/dense ratios and I_{NLL} (nats/token). Bigrams recur within 256 tokens (local), only beyond 256 (distant), or are novel (Introduction). 695 random 8192-token windows, three seeds, sampling nearly a validation set's worth of tokens, distinct from Table 1's tiling; 1024/4096 rows (24 windows): Table 21. At 8192, standard errors over evaluation batches are 0.0004–0.0014 per class and seed; they describe sampling of windows from this validation set, not of books. Dashes: unavailable. Dense-only values, token shares, and long-context sensitivity: Appendix D; rerun provenance: Appendix F.

Context (tokens)	class	dense	windowed	ratio	I_{NLL}
8192	novel bigram	$+0.0082 \pm 0.0014$	$\mathbf{+0.0238 \pm 0.0016}$	$\mathbf{2.9}\times$	$+0.0156 \pm 0.0022$
	distant recurrence	$+0.0022 \pm 0.0059$	$+0.0186 \pm 0.0022$	—	$+0.0164 \pm 0.0043$
	local recurrence	-0.0009 ± 0.0048	$+0.0156 \pm 0.0054$	sign flip	$+0.0165 \pm 0.0093$

5.5 The extra gain spans bigram-recurrence classes

To locate the extra windowed gain, we ask which tokens' predictions it improves. The natural explanation is that the convolution compensates for what the window removes: under a 256-token window, most layers can no longer see a repetition from further back, so the convolution should help mainly on tokens that depend on distant context. We test this by splitting the paired per-token loss reductions on PG-19 by whether the target token's bigram (the token and its predecessor) appeared earlier in the context: within the last 256 tokens (local recurrence, information the window keeps), only further back (distant recurrence, information the window removes), or not at all (novel bigram), following the associative-recall account of Arora et al. (2023).

The extra gain is not confined to where the window removes information. At 8192 tokens, windowing adds about 0.016 nats of improvement per token, roughly 1.6% in perplexity, to each class: distant recurrences, local recurrences that the window leaves visible, and novel bigrams (Table 5, last column). These are similar point estimates, not a demonstrated equality: across seeds, the paired difference between the local-recurrence and novel-bigram interactions has a 95% t-interval of $[-0.027, +0.029]$ nats, wide relative to the interaction itself. Each class contributes roughly in proportion to its share of tokens.

Under dense attention the picture differs. The smaller dense gain sits mostly on novel bigrams, the only class whose reduction is positive in every context and seed; they carry 81–101% of the dense loss reduction. The recurrence-class reductions are near zero or have standard deviations as large as their means (Appendix D).

Bigram recurrence is only a proxy for needing distant context, so we repeat the split using a direct measure: how much each token's loss improves when the model sees the full context rather than a window (Fang et al., 2025), measured by an evaluator model. Here the picture is mixed. Per token, the interaction grows with long-context need: tokens gaining more than 0.5 nats from full context make up 5.5% of tokens and carry 25% of the interaction, about $4.5\times$ their share, and the top bin has $12\times$ the per-token interaction of the lowest. But these tokens are few, so about 70% of the interaction falls on the 88% of tokens that gain at most 0.5 nats. On LongPPL's own key tokens, the convolution recovers only 36% of the window penalty (Appendix D).

Together these do not support an explanation in which the convolution only restores what the window cuts off: it helps tokens the window hurts most, per token, but it also improves predictions broadly, including ones the window leaves untouched. Aggregate perplexity recovery therefore does not imply recovered long-context ability. The broad component fits the controls in § 5.4, where a branch that mixes no neighbors keeps most of the gain. Appendix D checks that the pattern is not an artifact of the weaker windowed baseline.

6 Discussion

Under paired-seed attribution, the dense-attention gain replicates across seeds and contexts; in 81M models the windowed gain at long context is $3.5\times$ as large. The extra gain descriptively tracks the window penalty and shrinks as training lengthens (Table 12).

The hybrid result limits the recipe's scope because the attention-slot branch adds little when the restricted mixer incurs no penalty and cannot replace recurrent-block mixing (Table 16). It offers a low-cost benefit for windowed attention at long contexts, rather than a general addition. This boundary distinguishes the added attention-slot branch from built-in convolutions. Larger gains reported for linear attention and state-space blocks than for dense attention (Allen-Zhu, 2025; Gu & Dao, 2023) are consistent with benefit increasing with the mixer's baseline deficit; our experiment tests the added branch at specific sites.

A common intuition for the amplification is substitution: with enough context, dense attention learns the branch's job, while a window prevents it. A registered test of this intuition through the trained branch's amplitude sensitivity fails: sensitivity falls rather than rises with context under windowing, and restricting a dense model's attention lowers it (Appendix E). Because dependence on a branch and curvature under rescaling it are different quantities, the broader substitution account is untested rather than refuted.

The controls in §§ 5.4–5.5 also narrow the explanation. Most of the windowed gain does not require neighbor mixing; in one seed, most of the advantage survives removing the branch after 60% of training followed by further training; the gain is broad across token classes while being largest per token where the window hurts most.

One interpretation consistent with these results, among others we have not excluded, is that the zero-initialized branch changes optimization: the baseline uses attention capacity to learn what the branch supplies from the first update, and windowing may prolong the period in which that input is useful. Replicated switches at several training steps, windowed insertion, separating the shared weights from optimizer state, and tracking local-head formation with and without the branch would test it.

The practical comparison combines recovered quality with measured training time. For 81M models on PG-19 at an 8192-token context and this budget, the results support a $W{=}256$ attention window with the added branch. In logged training throughput on $2{\times}$A100, adding the branch lengthens steps by 12–13% at this context. Windowed attention with convolution has 15% higher throughput (13% shorter steps) than dense attention with convolution, but only 1.6% higher than the plain dense baseline (1.3–1.9% across seeds; Table 14). Most of the branch's cost is implementation: computed as four shifted multiply-adds and compiled, it adds 4% per step on A100, and a windowed step with it is then 11% shorter than a dense-baseline step (Table 15).

7 Limitations

The study covers small models from one architecture family and tokenizer, with only one pair at the larger scale; the amplification is untested beyond 81M, since the 253M pair tests only the dense gain, although paired-seed attribution transfers unchanged. The windowed results come from one long-document corpus, one principal budget, and one mixed layout (eight windowed and two dense layers), and measure aggregate perplexity rather than long-context capability. Several auxiliary experiments have limited seed replication; three runs were resumed and completed outside strict paired-seed attribution (Appendix F), and the interaction's mechanism remains an open question.

Paired-seed attribution does not remove execution nondeterminism or hardware and precision differences, so we compare effects within a training stack and treat small single-pair differences cautiously; Appendix F.1 reports the repeatability measurements and the threshold used.

8 Conclusion

Paired-seed attribution resolves a modest benefit from a zero-initialized convolution before attention and, in 81M models, a larger one under windowed attention at long context. Across the tested conditions, the extra benefit descriptively tracks the baseline cost of windowing and decreases with longer training in the budget experiment. A pointwise branch recovers most of the windowed gain, and in one seed mid-training removal preserves most of it, which separates the practical gain from the local-mixing explanation; what causes the extra benefit under windowing remains open.

Ethics statement

This work studies training efficiency of small language models on public corpora (OpenWebText, PG-19); it introduces no new data collection, human subjects, or foreseeable dual-use concerns beyond those of language modeling generally.

Broader impact statement

Cheaper local structure may reduce the compute (and associated energy) needed to reach a given perplexity; our results also clarify which architectural additions justify their complexity; we do not anticipate negative impacts specific to this method.

Reproducibility statement

Code is available at an anonymized repository, `https://anonymous.4open.science/r/Structure-Before-Attention-E24C`. Appendix A specifies the initialization and gradient requirements. The repository provides all training configurations, cluster manifests, exact-equality tests enforcing P1–P2, and the scripts behind every result. For regenerating the reported numbers without rerunning training, it includes a full-precision manifest of every deterministically rescored endpoint with the rescoring logs, the raw logs of the probes, and an export of the tracker histories; the included collector recomputes the final-results and paired-delta tables from that export offline. Checkpoints are not included because of their size.

Statement on the use of generative AI

We used generative AI tools in this work to design experiments and give feedback on the methodology, to propose and refine hypotheses (including the attention-substitution account and its preregistered tests), to implement the models, training and evaluation code, data preprocessing, and cluster jobs, to support the statistical analysis, and to interpret results. We did not use generative AI to generate the data for any reported experiment (a synthetic token stream is used only in software tests), and the remaining required-disclosure tasks (theoretical models, mathematical claims or proofs, translation) do not apply to this work. We also used these tools to write code, produce figures, search and summarize the literature, brainstorm candidate modules, draft and edit the text, and check the bibliography.

We have reviewed all AI-assisted work. Reported numbers were computed by scripts from the raw training and evaluation logs and cross-checked against them, and the headline results were rederived independently by deterministic rescoring of saved checkpoints. Hypotheses developed with AI assistance were tested against predictions fixed before the test runs completed (Appendix G); several failed and are reported as failures. Citations were checked against publisher and arXiv records with a citation-verification tool and corrected by hand. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

References

Zeyuan Allen-Zhu. Physics of language models: Part 4.1, architecture design and the magic of Canon layers. In *Advances in Neural Information Processing Systems 38*, pp. 47319–47339, 2025. doi: 10.52202/085713-1410.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. *arXiv preprint arXiv:2312.04927*, 2023. doi: 10.48550/arXiv.2312.04927.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian McAuley. ReZero is all you need: Fast convergence at large depth. In *Conference on Uncertainty in Artificial Intelligence*, 2021. arXiv:2003.04887.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Tal Arbel, Chris Pal, Gaël Varoquaux, and Pascal Vincent. Accounting for variance in machine learning benchmarks. In *Proceedings of Machine Learning and Systems (MLSys)*, 2021.

Loïc Cabannes, Maximilian Beck, Gergely Szilvasy, Matthijs Douze, Maria Lomeli, Jade Copet, Pierre-Emmanuel Mazaré, Gabriel Synnaeve, and Hervé Jégou. Short window attention enables long-term memorization. *arXiv preprint arXiv:2509.24552*, 2025. doi: 10.48550/arXiv.2509.24552.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2Net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*, 2016.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning*, 2024. arXiv:2405.21060.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, S. Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, 2017.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2411.13676. arXiv:2411.13676.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. What is wrong with perplexity for long-context language modeling? In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.23771. arXiv:2410.23771.

Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. In *International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2212.14052. arXiv:2212.14052.

Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024. doi: 10.48550/arXiv.2408.00118.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023. doi: 10.48550/arXiv.2310.06825.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-Mamba language model. *arXiv preprint arXiv:2403.19887*, 2024. doi: 10.48550/arXiv.2403.19887.

NVIDIA. Nemotron-H: A family of accurate and efficient hybrid Mamba-transformer models. *arXiv preprint arXiv:2504.03624*, 2025.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova Dassarma, T. Henighan, Benjamin Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022. doi: 10.48550/arXiv.2209.11895.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28043–28078, 2023.

Ziqing Qiao, Yinuo Xu, Chaojun Xiao, Zhou Su, Zihan Zhou, Yingfa Chen, Xiaoyue Xu, Xu Han, and Zhiyuan Liu. Rethinking the role of efficient attention in hybrid architectures. *arXiv preprint arXiv:2606.15378*, 2026. doi: 10.48550/arXiv.2606.15378.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. arXiv:1911.05507; introduces PG-19.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2406.07522. arXiv:2406.07522.

Oliver Sieberling, Bharat Runwal, Rameswar Panda, and Yoon Kim. Dynamic short convolutions improve transformers. *arXiv preprint arXiv:2606.03825*, 2026. doi: 10.48550/arXiv.2606.03825.

David R. So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V. Le. Primer: Searching for efficient transformers for language modeling. In *Advances in Neural Information Processing Systems*, 2021. arXiv:2109.08668.

Yuchuan Tian, Yingte Shu, Wei He, Shuo Zhang, Tianchen Zhao, Chao Xu, Xinghao Chen, Yunhe Wang, Hanting Chen, and Yu Wang. Convolution for large language models. *arXiv preprint arXiv:2607.18413*, 2026. doi: 10.48550/arXiv.2607.18413.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, 2019. doi: 10.18653/v1/P19-1580.

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2019. arXiv:1901.10430.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2412.06464. arXiv:2412.06464.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *IEEE International Conference on Computer Vision*, pp. 3813–3824, 2023. doi: 10.1109/ICCV51070.2023.00355. arXiv:2302.05543.

A Paired-seed attribution implementation: α-gradient equalization and random-stream discipline

This appendix records the implementation contract that makes the paired protocol exact, and the initialization detail that makes the candidate modules comparable to one another. Both are enforced by tests in the released code.

PER-SITE GENERATORS KEEP THE GLOBAL RNG STREAM IDENTICAL

The protocol requires that inserting a branch changes nothing about the baseline run except the branch itself. Drawing the branch's interior weights from the global generator would violate this: subsequent random draws would change, affecting all shared weights and, through the data loader, the batch order. Each insertion site therefore builds its parameters inside `torch.random.fork_rng()`, from a dedicated generator seeded deterministically per site and per module class, $\texttt{seed} = \texttt{0x51C0DE} + (10^6{+}3) \times \texttt{salt} + \texttt{site_idx}$, where `salt` identifies the module class and `site_idx` its insertion site. The global stream is left byte-identical to the baseline's. A regression test checks that model construction leaves the global generator in the same state with and without the branch. It also checks that paired runs produce identical batch sequences. This caught an earlier depthwise convolution implementation that consumed global randomness through the default `nn.Conv1d` initializer.

IDENTITY AT INITIALIZATION, TWO WAYS

The convolution satisfies the identity-at-initialization contract by initializing its taps directly to zero. The α-gated candidates instead use a ReZero-style outer gate, $x + \alpha \odot \mathrm{Branch}(\mathrm{RMSNorm}(x))$ with $\alpha \in \mathbb{R}^d$ zero-initialized, which is exactly the identity at step 0 without zeroing interior weights. The dynamic-tap convolution deliberately uses a zero-initialized controller projection, Q_{tap}, mapping controller features to tap modulations and distinct from the attention query projection. Learning proceeds in three stages: first α, then the taps and Q_{tap}, and finally the tap-mixing matrix once Q_{tap} becomes nonzero. Until then, the matrix's gradient is identically zero.

The staging is documented and tested rather than incidental.

EQUALIZING α'S STEP-ZERO GRADIENT

At step 0, a zero gate suppresses every interior gradient, so only α changes. Its gradient, $\partial \mathcal{L} / \partial \alpha$, scales with branch-output RMS and therefore sets how quickly learning begins inside the branch. Under a shared learning rate, modules whose branch outputs differ greatly in scale unlock at different speeds. Without controlling this scale, comparisons conflate initialization with function class.

The bilinear operator made this concrete: with its output projection initialized at a fixed 0.02, α's step-zero gradient was roughly $33\times$ smaller than the sigmoid-gated convolution's, because the rank-k contraction shrinks the branch output. Initializing that projection at $k^{-1/2}$ instead restores the branch RMS to the same decade (the interior projections use the usual $d^{-1/2}$). A test asserts that the three candidate modules' α gradients agree to within a factor of 12 at step 0 (a coarse scale check). This matters for interpretation: the finding that the bilinear operator ties the convolution (§ 5.4) is only meaningful once both unlock at comparable speed.

GLOBAL-INITIALIZATION GUARD

The model applies a global weight initializer after construction, which would overwrite the carefully seeded branch parameters. The branch's linear layers are tagged `_skip_global_init` and skipped. This was caught by a test asserting branch weights are unchanged after the global pass, not by inspection.

BOTH RUNS OF EACH PAIR

Figure 2 plots the two runs of each dense pair in the OpenWebText 1024-token cell, seeds 42–44, from the tracker's validation histories. The baselines from different seeds end 2.3% apart in training-time validation, which draws different validation windows per seed, more than twice the effect being measured, and their curves cross repeatedly during training (panels a–b). Within a seed, the baseline and the +conv run follow the same trajectory and the +conv run stays below it; after step 10k the paired gain never falls below 0.49% in any seed (panel c). Comparing a baseline with the +conv run of a *different* seed, which is what an unpaired study does, gives final gains from -1.2% to $+2.7\%$ across the six cross-seed combinations, with the wrong sign in two of them. Step-zero equality (P1) is checked by the tests above rather than plotted, since validation is first logged at step 1,000.

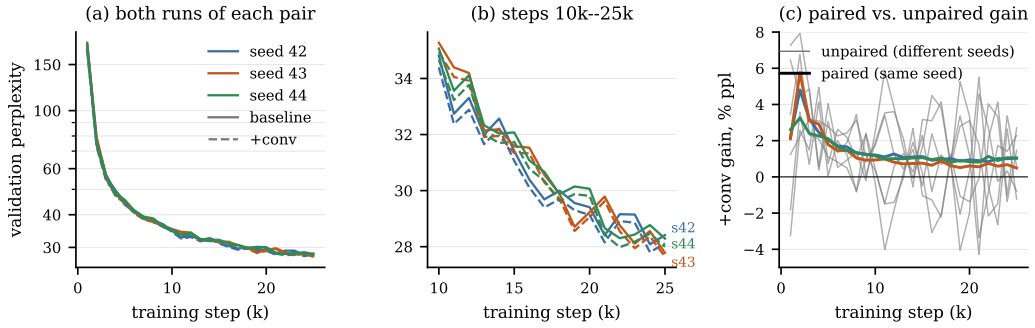


Figure 2: Validation perplexity of both runs in each paired-seed attribution pair (OpenWebText, 1024 tokens, dense attention, seeds 42–44). (a) Full training; solid lines are baselines and dashed lines the +conv runs, colored by seed. (b) Steps 10k–25k. (c) Relative gain $(\mathrm{ppl}_B - \mathrm{ppl}_S)/\mathrm{ppl}_B$: colored lines pair each baseline with the +conv run of the same seed; grey lines pair it with the +conv run of a different seed. Tracker perplexities on sampled validation batches.

B Additional Experimental Results

This appendix supplies detailed comparisons behind §§ 5.1–5.4: the per-seed factorial, window and budget comparisons, and branch controls. The experiments span 81M–253M parameters and approximately 1.6–5.2B training tokens (the larger run is also rounded to 5.2B); exact budgets appear in Table 1. Prior work considers convolutions with widths of 3–4. A 256-token window at 8192 tokens reduces a windowed layer's attention computation by more than an order of magnitude, while the book-text baseline penalty exceeds 1% (Table 3).

Table 6: OpenWebText factorial per seed (paired deltas; final validation perplexities where tabulated). Windowed attention with convolution beats the dense *baseline* in every configuration; at 4096, within seeds 42 and 43 it is the best of the four configurations; for seed 44, dense attention with convolution is marginally best.

Context (tokens)	seed	dense base	dense+conv	windowed base	windowed+conv	Δ(windowed)	I_{ppl} (pp)
1024	42	28.41	28.12	28.47	28.20	+0.93%	−0.11 pp
1024	43	27.78	27.64	27.94	27.64	+1.09%	+0.60 pp
1024	44	28.29	28.00	28.38	28.05	+1.17%	+0.13 pp
4096	42	27.10	26.90	27.09	26.83	+0.94%	+0.23 pp
4096	43	27.50	27.24	27.42	27.12	+1.08%	+0.16 pp
4096	44	27.51	27.26	27.64	27.30	+1.22%	+0.32 pp

The initial window sweep and its replications provide descriptive comparisons rather than an established optimal window (Table 11). Figure 1 uses three factorial seeds at $W{=}256$, four measurements at $W{=}64$ and 512 (seed 43 enters twice, once per stack) and three at $W{=}128$ and 1024, from the replications in Tables 9 and 10; all but $W{=}256$ pool two training stacks, so n counts measurements rather than independent seeds. Figure 3 separates the stacks.

B.1 Paired Learning Curves

The endpoint comparisons leave open when the interaction emerges and whether it persists through training. The paired curves show that trajectory. Figure 4 shows paired PG-19 configurations across training steps using tracker histories and shared sampled validation batches. These perplexities are lower than deterministic rescoring. The seed-43 windowed baseline and seed-44 windowed convolution histories at 8192 tokens end at step 24,000; Appendix F explains why.

Three features recur in each paired configuration. The paired gain peaks at steps 2–4k and approaches a plateau by roughly step 10k. The windowed plateau exceeds the dense plateau by more at longer contexts. After the transient, interaction and window penalty move together, nearly coincide at 8192 tokens, and both shrink at the $1.8\times$ budget.

Table 7: PG-19 factorial per seed (training-time final validation perplexity of the original runs; deltas from unrounded values). Rows marked ‡ are the deterministic rescoring of the same checkpoints on non-overlapping tilings of the whole validation set (at 8192, seed-42 windowed pair: the rerun); these are the rows Table 3 summarizes. P_{base}, I_{ppl}, and P_{conv} use Table 3's conventions. Values marked † were read from the training log rather than the experiment tracker (Appendix F); the log gives them to two decimals, so entries derived from them carry up to ± 0.03 pp of rounding.

Context (tokens)	seed	dense base	dense+conv	windowed base	windowed+conv	P_{base}	I_{ppl}	P_{conv}
1024	42	19.73	19.60	19.90	19.72	$+0.85\%$	$+0.25$ pp	$+0.60\%$
1024	43	20.15	20.01	20.29	20.13	$+0.71\%$	$+0.10$ pp	$+0.60\%$
1024	44	19.85	19.69	19.95	19.77	$+0.50\%$	$+0.09$ pp	$+0.40\%$
4096	42	19.51	19.26	19.68	19.39	$+0.90\%$	$+0.22$ pp	$+0.67\%$
4096	43	18.89	18.75	18.92	18.71	$+0.13\%$	$+0.33$ pp	-0.20%
4096	44	18.60	18.47	18.83	18.53	$+1.27\%$	$+0.92$ pp	$+0.33\%$
8192	42	18.59	18.49	18.89†	18.48	$+1.64\%$	$+1.67$ pp	-0.06%
8192	43	19.23	19.12	19.51†	19.10	$+1.47\%$	$+1.51$ pp	-0.07%
8192	44	19.11	18.99	19.36	18.96†	$+1.30\%$	$+1.42$ pp	-0.15%
8192‡	42	18.99	18.90	19.29	18.90	$+1.60\%$	$+1.54$ pp	$+0.03\%$
8192‡	43	18.93	18.81	19.20	18.80	$+1.45\%$	$+1.46$ pp	-0.04%
8192‡	44	19.00	18.87	19.25	18.84	$+1.32\%$	$+1.45$ pp	-0.16%
4096‡	42	18.85	18.62	19.03	18.74	$+0.93\%$	$+0.27$ pp	$+0.65\%$
4096‡	43	18.90	18.75	18.93	18.72	$+0.15\%$	$+0.34$ pp	-0.19%
4096‡	44	18.83	18.70	19.07	18.76	$+1.27\%$	$+0.94$ pp	$+0.31\%$
1024‡	42	19.55	19.41	19.70	19.53	$+0.73\%$	$+0.12$ pp	$+0.61\%$
1024‡	43	19.54	19.39	19.64	19.48	$+0.50\%$	$+0.02$ pp	$+0.47\%$
1024‡	44	19.58	19.43	19.66	19.47	$+0.40\%$	$+0.18$ pp	$+0.22\%$

Table 8: Window-size sweep at an 8192-token context on PG-19 (seed 42, deterministic rescoring). $P_{\text{base}}(W)$ is baseline penalty (%); $I_{\text{ppl}}(W)$ is interaction (pp), with dense baseline 18.99 and dense convolution gain $+0.49\%$. Single seed; duplicate runs differ by $\sim 0.17\%$ (~ 0.3 pp on a difference of deltas).

W	windowed base	windowed+conv	Δ (%)	$P_{\text{base}}(W)$	$I_{\text{ppl}}(W)$	versus dense+conv
64	19.04	19.02	$+0.08\%$	$+0.23\%$	-0.41 pp	$+0.65\%$
128	19.20	18.94	$+1.34\%$	$+1.06\%$	$+0.85$ pp	$+0.19\%$
256	19.30	18.91	$+2.03\%$	$+1.60\%$	$+1.54$ pp	$+0.03\%$
512	19.15	**18.84**	$+1.63\%$	$+0.83\%$	$+1.14$ pp	$\mathbf{-0.33\%}$
1024	19.17	18.92	$+1.33\%$	$+0.95\%$	$+0.84$ pp	$+0.09\%$

B.2 Throughput and branch controls

The logged overhead of Table 14 reflects the branch as trained, in eager PyTorch. A controlled single-GPU benchmark (Table 15) times full training steps of the 81M model at 8192 tokens, with every variant interleaved on one GPU and the median of seven rounds reported. On A100 it reproduces the logged overhead ($+13\%$ dense, $+15\%$ windowed, as trained) and the logged dense-baseline throughput (92.5k tokens/s per GPU). The width-one branch computes $x + w \odot \mathrm{RMSNorm}_{\gamma}(x)$, which equals $x + \mathrm{RMSNorm}_{\gamma \odot w}(x)$, one normalization with folded weights; this fused form matches the trained operator's logits to within $3 \cdot 10^{-6}$, as does the width-four convolution computed as four shifted multiply-adds. Compiling only the branch with `torch.compile` roughly halves the overhead; the compiled shifted-add width-four branch costs 4% per step on A100 and the fused width-one branch 1–1.5%. Most of the measured overhead is therefore implementation rather than arithmetic. With the compiled width-four branch, a windowed step takes 157 ms against 177 ms for the dense baseline on the same GPU.

Candidate operators

The rank-16 bilinear operator multiplies projections of current and previous token representations across channels; the convolution combines more positions. We do not claim either is formally more expressive. The dynamic-tap convolution uses a small controller to modulate its taps from the current

Table 9: Paired gains and window penalties for $W{=}64$ and $W{=}512$, grouped by seed and training stack: seeds 43/44 on V100/fp16 and seed 43 on A100/bf16. Deterministic rescoring uses the nonoverlapping validation tiling at an 8192-token PG-19 context and the pilot budget. Comparisons use dense pairs trained on the same stack. Seed-43 dense gains differ by $0.58\,\mathrm{pp}$ between stacks, so absolute perplexities are comparable only within a stack. Negative last-column values favor windowed attention with convolution.

seed	Δ(dense)	$P_{\mathrm{base}}(64)$	Δ(windowed, 64)	$P_{\mathrm{base}}(512)$	Δ(windowed, 512)	windowed+conv(512) versus dense+conv
43	$+1.21\%$	$+0.65\%$	$+0.54\%$	$+0.54\%$	$+1.19\%$	$+0.56\%$
44	$+0.56\%$	$+0.05\%$	$+1.01\%$	$+0.50\%$	$+1.51\%$	-0.45%
43 (A100)	$+0.63\%$	$+0.90\%$	$+0.71\%$	$+0.83\%$	$+0.75\%$	$+0.71\%$

Table 10: Paired gains, penalties, and interactions for $W{=}128$ and $W{=}1024$, seeds 43/44 on V100/fp16, against dense pairs on that stack. Deterministic rescoring uses the nonoverlapping validation tiling at an 8192-token PG-19 context and the pilot budget. Negative last-column values favor windowed attention with convolution; Appendix F documents resumed training.

seed	Δ(dense)	$P_{\mathrm{base}}(128)$	Δ(windowed, 128)	$I_{\mathrm{ppl}}(128)$	$P_{\mathrm{base}}(1024)$	Δ(windowed, 1024)	$I_{\mathrm{ppl}}(1024)$	windowed+conv(1024) versus dense+conv
43	$+1.20\%$	$+0.58\%$	$+1.17\%$	$-0.04\,\mathrm{pp}$	$+0.03\%$	$+0.33\%$	$-0.87\,\mathrm{pp}$	$+0.91\%$
44	$+0.56\%$	$+1.03\%$	$+1.54\%$	$+0.98\,\mathrm{pp}$	-0.01%	$+1.27\%$	$+0.71\,\mathrm{pp}$	-0.73%

token representation and reaches a similar result under dense attention. Under windowed attention at 8192 tokens, none of the three α-gated operators shares the convolution's amplification (Table 18), so the cheap local composition all four supply can explain at most the dense gain. The sigmoid-gated convolution multiplies output by a token-dependent σ, distinct from α. Its smaller gain fits that interpretation if gating suppresses delivery, which we did not test. For α-gated candidates, step-zero gradient equalization coarsely controls how quickly the gate opens and permits interior learning; gradients agree within a factor of 12 (Appendix A).

The pointwise control is reported in § 5.4 and Table 4.

C Prior results on short convolutions

Table 19 places our measurements beside the closest published comparisons. Setups differ in scale, data, placement, and initialization, so the gains are not directly comparable; the table records what each study varied.

D Token-level decompositions

Dense-configuration decomposition over all three seeds

Table 20 gives the dense-attention decomposition with token shares and each class's share of the total loss reduction. Only the novel-bigram class has a positive reduction in every context and seed; the recurrence classes are near zero or have standard deviations as large as their means. The class labels are defined by exact-bigram recurrence, not by any latent computation: "no earlier occurrence" is a proxy for novel prediction, and recurrence is not evidence that a model copied rather than predicted. Sampled windows are not constrained to single books, so class shares carry a small boundary-contamination bias that is identical for both members of each pair.

Relative-improvement check

Absolute reductions could appear larger under windowing simply because starting losses are higher. We check whether the class pattern persists after normalization by baseline loss. On the 695-window random panel at 8192 tokens, novel-bigram reductions are $+0.22\%$ under dense attention and $+0.63\%$ under windowed attention, retaining the $2.9\times$ ratio. Distant-recurrence reductions are $+0.10\%$ and $+0.86\%$. Local recurrence retains its sign change, from -0.11% to $+1.85\%$.

The local-recurrence amplification cautions against interpreting the interaction as compensation for information removed by the window. These bigrams recur within the window and should be least

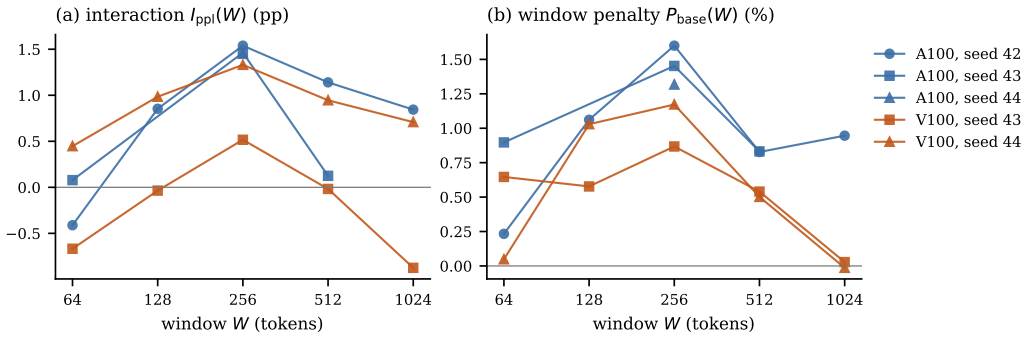


Figure 3: Window-size sweep at 8192 tokens on PG-19, by training stack (A100/bf16, V100/fp16) and seed, deterministic rescoring. Each point compares windowed models with a dense pair trained on the same stack; lines join one seed within one stack. (a) Interaction $I_{\text{ppl}}(W)$; (b) window penalty $P_{\text{base}}(W)$. Data: Tables 8–10 and the W=256 pairs of Tables 7 and 4.

Table 11: Summary of window-size sweep results and replications across seeds and training stacks; individual measurements appear in Tables 8 and 9.

Comparison	Measurement
Seed-42 initial sweep	$W \in \{64, 128, 256, 512, 1024\}$; $r(P_{\text{base}}, I_{\text{ppl}}) = 0.92$ across five points. Both penalty and interaction peak at W=256 and reach their minimum at W=64.
Seeds 43/44, V100/fp16	Penalties: $+0.35 \pm 0.42\%$ at 64, $+0.52 \pm 0.03\%$ at 512; interactions: -0.11 ± 0.79 pp and $+0.46 \pm 0.68$ pp.
W=512 versus dense+conv	One replication ahead, the other behind (approximately -0.5% and $+0.5\%$, respectively).
Seed 43, A100/bf16	Interactions: $+0.08$ pp at 64 and $+0.12$ pp at 512.
Seeds 43/44, V100/fp16, W=128/1024	Penalties: $+0.80 \pm 0.32\%$ at 128, $+0.01 \pm 0.03\%$ at 1024; interactions: $+0.47 \pm 0.72$ pp and -0.08 ± 1.12 pp (Table 10).

affected by windowing. Nevertheless, their gain changes from -0.0009 ± 0.0048 under dense attention to $+0.0156 \pm 0.0054$ under windowed attention at 8192 tokens.

Decomposition by long-context sensitivity

Recurrence classes leave open whether amplification concentrates on tokens that benefit from long context. We address that question using measured loss changes under shorter histories. Following LongPPL (Fang et al., 2025), we score eligible tokens of the 8192-token PG-19 validation tiling from Table 1 (695 windows, 5.69M tokens, distinct from the random token-class panel) by the long–short difference $\text{LSD} = \text{NLL}_K - \text{NLL}_{8192}$, the loss saved by extending a short context to the full window. The strided short-context protocol (K=256; chunks of $2K$, stride K) gives each scored token between 256 and 511 tokens of history, and the first $2K$ positions of each window are excluded. Using the same model to define bins and estimate contrasts creates selection bias. Selecting low dense-model LSD selects lucky tokens and produced a spurious negative window penalty in the largest bin. We therefore *cross-fit*: each seed's bins use the mean of the other seeds' dense baselines.

Table 22 reports, per bin, the window penalty, the convolution's gain under each attention type, and the interaction.

The window penalty is concentrated on tokens with high measured long-context sensitivity. Per-token penalties rise monotonically across bins, from 0.012 in the 0–0.5 bin to 0.368 above 2, a 32× increase. Tokens with LSD > 0.5 constitute 5.5% of the total and carry 63% of the penalty. The stricter preregistered majority threshold above 1 narrowly fails: those 2.2% of tokens carry 43%.

Table 12: Changes in window penalty, dense-attention gain, and interaction with longer training at $W{=}256$, PG-19, seed 42, using deterministic rescoring. The longer budget is 45k steps and 2.95B sampled tokens, one corpus-equivalent (approximately as many sampled tokens as the corpus contains, without requiring each token to be seen once). Appendix F documents resumed training. Dashes denote unavailable entries.

Measurement	Original budget	1.8× budget
Window penalty	+1.60%	+0.78%
Dense convolution gain	+0.49%	+0.85%
Interaction	+1.54 pp	+0.94 pp
Interaction-to-penalty ratio (fraction)	0.96	1.21
windowed+conv versus dense+conv perplexity	—	17.09 versus 17.12

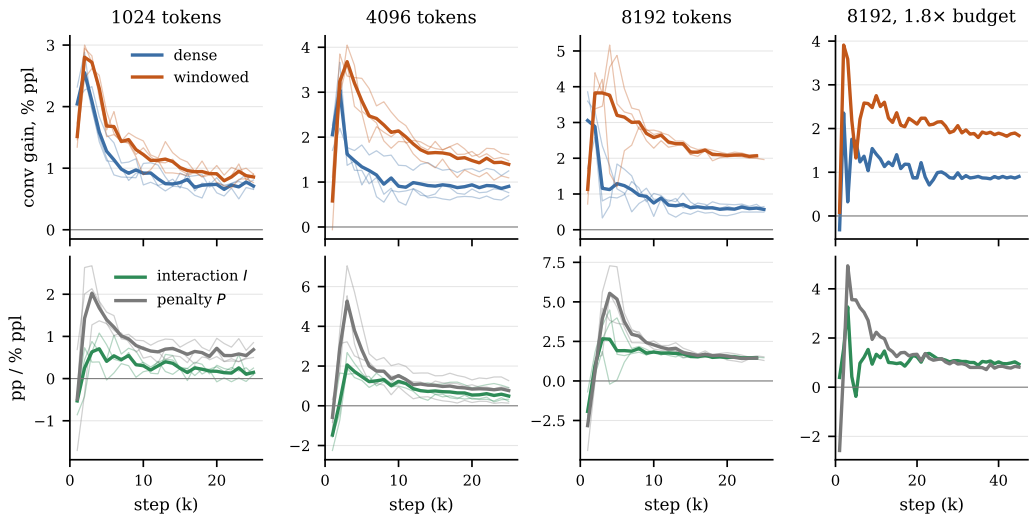


Figure 4: Per-step paired deltas on PG-19. Top: paired convolution gain (base−convolution, % perplexity) under dense and windowed attention; thin lines are seeds 42–44, thick lines their mean (seed 42 only at 1.8×). Bottom: the interaction I_{ppl} (windowed minus dense gain, pp) and the window penalty P_{base} (%). The plot abbreviates these as I and P.

The convolution's extra gain under windowing is largest per token in the tail: the gain above LSD 2 is 12× that at LSD $\leq$ 0. Under dense attention, the aggregate contribution above LSD 0.5 is approximately zero. Binwise estimates there are noisy, reaching ± 0.05 in the highest bin. To that extent the convolution under windowing helps on exactly the tokens the window hurts.

That component nevertheless accounts for a minority of the interaction. About 70% lies in the two bins with LSD $\leq$ 0.5, containing 88% of tokens; the share rises to 79% at a 1.0-nat threshold. One-third lies in the LSD $\leq$ 0 bin, where the mean window penalty is indistinguishable from zero (-0.001 ± 0.002) and the windowed gain is 2.8–3.2× the dense gain.

Low LSD is an evaluator-defined proxy, not an architectural property, so this does not show that the windowed model is insensitive to long context on those tokens. The broad pattern recurs at $K{=}1024$. On LongPPL's own key set (LSD $>$ 2, long-context NLL $<$ 2; 0.56% of tokens) the window costs 0.226 nats per token, of which the convolution recovers 36% under the window; under dense attention its gain there is -0.009 ± 0.041.

Earlier interpretations not sustained by replication

An earlier 24-window, two-seed sample suggested a 1.5× enrichment of the interaction on novel-bigram tokens. The 695-window random panel does not reproduce it: those tokens carry 64% of the interaction, matching their token share.

Table 13: Final perplexity after transplanting final-budget taps or inserting/removing the branch at step 15k of 25k; seed 42, 8192 tokens, one stack, matched rescoring. Resumed training: Appendix F.

Experiment	Configuration	Final perplexity
Windowed transplant	Zero-initialized taps	18.9034
	Own endpoint taps, frozen	18.9862
	Own endpoint taps, trainable	18.9868
	Dense endpoint taps, frozen	19.0189
Windowed removal	Baseline sham	19.293
	Branch removed at 15k	19.004
	Intact endpoint	18.903
Dense insertion	Baseline continuation	18.991
	Branch inserted at 15k	18.969
	Branch present from start	18.888

Own-tap transplants are 0.44% worse than zero initialization and frozen dense taps 0.61% worse; under this rescoring the own-tap gap (0.083 of the 0.39 perplexity gain) lies between the 0.06 equivalence and 0.10 materiality margins ($+0.10$ on the registered training-time metric). Switches use $W{=}256$.

Table 14: Logged training throughput of the four 8192-token PG-19 configurations on 2×A100-80GB (bf16, FlexAttention for the windowed configurations): per run, the median over logged steps after warm-up; mean ± SD over seeds 42–44, summed over both GPUs (the training log records one GPU's share). Relative step time is the dense baseline's throughput divided by each configuration's, with its per-seed range. Runs were not interleaved on one node; Table 15 gives controlled timings and peak memory.

	dense base	dense+conv	windowed base	windowed+conv
tokens/s (10^3)	185.1 ± 0.2	164.1 ± 0.7	210.2 ± 9.9	188.1 ± 0.4
relative step time	1.000	1.128 (1.125–1.132)	0.881 (0.857–0.932)	0.984 (0.981–0.987)

An earlier version of this analysis, run only at seed 42, found a large *distant recurrence* effect at 4096 ($+0.0222$ NLL, a quarter of the total reduction there) and read it as a transient "retrieval assist." Under replication the effect is -0.0014 at seed 43 and $+0.0033$ at seed 44 (mean $+0.0080{\pm}0.0125$), and the elevated seed-42 dense delta it was invoked to explain ($+1.28\%$) is $+0.91 \pm 0.32\%$ over three seeds. We record the episode because it is the failure mode paired-seed attribution exists to prevent: a mechanistic story fitted to one seed at an effect size inside seed noise.

E Branch perturbations and attention statistics

This appendix first presents the amplitude test, then checks whether its unexpected direction depends on evaluated tokens, perturbation direction, or perturbation size.

The amplitude test

A common intuition for the amplification is substitution: given enough context, dense attention learns to do the branch's job itself, while a window in eight of ten layers prevents it. If so, a trained model's dependence on its branch should fall or stay flat with context under dense attention, rise with context under windowing, and rise when a dense model's attention is restricted at inference.

We measure local amplitude sensitivity with $C(\epsilon) = L(1-\epsilon) + L(1+\epsilon) - 2L(1)$, where $L(\lambda)$ is validation loss after multiplying the trained branch output by λ. Scaling a bias-free convolution's weights implements this intervention exactly. The symmetric difference cancels the first-order term and locally estimates $\epsilon^2 L''(1)$. It measures amplitude sensitivity, not sensitivity to coefficient shape (Dinh et al., 2017); full removal is unsuitable for measuring this local quantity (below).

The observed changes oppose the registered directions where a direction was registered: windowed sensitivity declines with context rather than rising, dense sensitivity stays roughly flat (consistent

Table 15: Step-time overhead of the pre-attention branch relative to no branch, and peak training-step memory: 81M model, forward and backward, batch 2×8192 tokens, one A100-SXM4-80GB (bf16) or V100-SXM2-32GB (fp16); windowed columns use $W{=}256$ with dense attention in every fourth layer. Median of seven interleaved rounds of ten steps (spread across rounds $\leq 1.2\%$). "As trained": eager PyTorch; "compiled": only the branch under `torch.compile`. "Shifted adds": the same width-four convolution computed as four shifted multiply-adds. Memory: peak above the resident weights, identical under both attention types. Branch weights are nonzero, so the branch does real work.

		dense attention		windowed attention		peak memory
GPU	branch	as trained	compiled	as trained	compiled	compiled
A100	width four	$+13.0\%$	$+6.3\%$	$+15.2\%$	$+7.4\%$	$+4.4\%$
	width four, shifted adds	$+18.9\%$	$+3.7\%$	$+22.1\%$	$+4.3\%$	$+5.9\%$
	width one ($k{=}1$ convolution)	$+11.9\%$	$+6.0\%$	$+13.9\%$	$+7.0\%$	$+4.4\%$
	width one, fused normalization	$+5.8\%$	$+1.2\%$	$+6.7\%$	$+1.5\%$	$+2.9\%$
	no branch	177.2 ms	177.2 ms	150.9 ms	150.8 ms	13.4 GiB
V100	width four	$+7.4\%$	$+3.9\%$	$+5.1\%$	$+2.7\%$	—
	width one ($k{=}1$ convolution)	$+7.9\%$	$+4.8\%$	$+5.5\%$	$+3.3\%$	—
	width one, fused normalization	$+3.6\%$	$+0.6\%$	$+2.5\%$	$+0.5\%$	—
	no branch	528.6 ms	528.8 ms	766.3 ms	766.4 ms	—

Table 16: Attention–SSM hybrid factorial (8192-token PG-19, pilot budget (25,000 steps, 1.638B training tokens), V100/fp16, deterministic rescoring). Hybrid: layers 3 and 7 dense attention, the other eight Mamba-2 mixers (internal width expansion factor 1.5, state dimension $d_{\text{state}}{=}64$, parameters within 3% of the dense model). The dense control carries the convolution on the same two sites. P_{ssm} is the hybrid base's perplexity relative to the dense base.

seed	dense base	dense+conv$_2$	hybrid base	hybrid+conv	P_{ssm}	Δ(H)	Δ(D$_2$)	I_{ppl} (pp)
42	18.92	18.92	18.64	18.62	-1.49%	$+0.14\%$	$+0.01\%$	$+0.13$ pp
43	18.98	18.96	18.53	18.50	-2.35%	$+0.18\%$	$+0.10\%$	$+0.08$ pp
44	19.02	19.00	18.64	18.61	-1.99%	$+0.16\%$	$+0.11\%$	$+0.06$ pp
native SSM convolution reduced to $d_{\text{conv}}{=}1$ (no temporal mixing), seed 42: hybrid base 19.63, hybrid+conv 19.59 ($+0.25\%$)								

with its registered clause), and restricting a dense model's attention span lowers sensitivity rather than raising it (Table 23).

The reversed trend raises a second question, namely what the endpoint sensitivity statistic captures; follow-ups on perturbation direction, cross-layer interactions, withdrawal adaptation, and fractional-rescaling displacement show that it reflects endpoint amplitude sensitivity rather than the branch's training-time benefit. Under its registered decision rule the hypothesis, as operationalized, is withdrawn. Substitution does not in fact entail these predictions: whether attention has taken over the branch's function and how sharply the loss responds to rescaling the branch are different quantities, and normalization and co-adaptation can reverse their ordering, as the matched-displacement controls show. The broader substitution account is therefore untested rather than refuted. The subsections below report coefficient and separate-layer interventions. They also report gradual branch withdrawal over 400 further training steps followed by 600 steps with the branch disabled. Matched-displacement controls use equal-RMS perturbations relative to the residual entering attention normalization. The mechanism of the training-time benefit remains unexplained.

WHY SECOND-ORDER, AND WHY NOT REMOVAL

The step-15k removal experiment (§ 5.4) is followed by 10k further training steps, so the model adapts to the branch's absence before evaluation. Removing the branch from a trained model at evaluation, with no adaptation, asks a different question. Removing the branch entirely, $\lambda{=}0$, increases perplexity by 2300 to 22000% because every downstream layer trained around the branch; it measures entanglement rather than marginal contribution. The symmetric difference cancels the first-order term, so the statistic estimates $\epsilon^2 L''(1)$ as above; $C(\epsilon)/\epsilon^2$ estimates local curvature. $C(0.10)/C(0.05)$ is 4.03–4.08 at every context, consistent with this quadratic scaling.

Table 17: Final perplexity and relative reduction for candidate operators and initialization controls under dense attention (81M, 1024-token context, same paired-seed attribution protocol and budget; seed 42 unless noted). Three operator classes fall within 0.03 perplexity; gating recovers less than half. Default-size random taps yield no gain; taps a tenth that size match zero initialization. Dashes denote unavailable entries.

module	class	final perplexity	Δ (%)
baseline	—	28.41	—
convolution	static linear position-mixing	28.12	+1.04%
bilinear operator (k=16)	multiplicative cross-channel	28.12	+1.03%
dynamic-tap convolution	content-selected linear	28.09	+1.15%
sigmoid-gated convolution	content-gated linear	28.29	+0.42%
seed-43 replication: convolution / bilinear operator		27.64 / 27.64	+0.49% / +0.48%
Initialization control (same setting on the V100/fp16 stack)			
baseline	—	28.36	—
convolution, zero initialization	—	28.14	+0.77%
convolution, default initialization, taps $U(\pm 0.5)$	—	28.42	−0.23%
convolution, default initialization $\times 0.1$, taps $U(\pm 0.05)$	—	28.14	+0.78%

Table 18: Operators under dense and windowed (W=256) attention at an 8192-token context: paired relative perplexity reductions and interaction, mean ± SD over seeds 42–44, V100/fp16, each arm paired with same-stack baselines and deterministically rescored. Last column: the operator's mean windowed gain as a share of the convolution's. The windowed baseline penalty on this stack is $+1.32 \pm 0.53\%$. The seed-42 dense bilinear run was resumed from its step-15k checkpoint (Appendix F).

operator	Δ(dense)	Δ(windowed)	I_{ppl} (pp)	share of conv. windowed gain
convolution	$+0.74 \pm 0.40\%$	$+1.92 \pm 0.21\%$	$+1.17 \pm 0.59$ pp	100%
bilinear (k=16)	$+0.76 \pm 0.13\%$	$+0.59 \pm 0.17\%$	-0.17 ± 0.23 pp	31%
dynamic-tap convolution	$+0.98 \pm 0.27\%$	$+1.24 \pm 0.04\%$	$+0.27 \pm 0.26$ pp	65%
sigmoid-gated convolution	$+0.33 \pm 0.23\%$	$+0.46 \pm 0.19\%$	$+0.14 \pm 0.42$ pp	24%

Amplitude sensitivity decreases monotonically with context under windowed attention and remains approximately constant under dense attention, contrary to the registered contrast. Restricting a dense model's span from full to 256 tokens also lowers sensitivity to 0.86–$0.93\times$ its original value across three seeds. The windowed and span-restriction predictions fail; the dense clause is met by a flat result (Table 23).

Three follow-ups distinguish endpoint sensitivity from training-time gain. Tap-coefficient perturbations yield comparable curvature (0.99–1.04). To test whether joint-layer effects explain the contrast, we scale branches independently: their responses sum to the same 0.55–0.65 ratio as joint scaling, in predictive KL (the sum of KL divergences from intact predictions to positively and negatively perturbed predictions, without target labels) and NLL curvature. End-of-training withdrawal (seed 42, 1k low-learning-rate steps) costs about the same in both configurations (0.023 versus 0.025 nats). Earlier removal at 15k with 10k adaptation steps keeps most of the advantage (§ 5.4).

Matched target population across contexts

The context sweep of Table 23 scores the first 128 windows at each length: that is, different books at 1024 and 8192 tokens. We repeat the measurement on the last 256 targets of 128 common 8192-token tiles, using checkpoints corresponding to 1024-, 4096-, and 8192-token histories. For seeds 43 and 44, predictive-KL ratios are $1.19 \to 0.71 \to 0.65$ and $0.95 \to 0.81 \to 0.56$. The trend therefore persists when evaluated text is held fixed.

Radial versus nonradial curvature

The intervention in the amplitude test above scales the branch, producing a purely *radial* movement in weight space: a bias-free convolution keeps its temporal response at every positive scale, so radial curvature estimates measure amplitude sensitivity and nothing about the filter's coefficient shape. We therefore repeated the measurement along random directions orthogonal to each channel's tap

Table 19: Short causal convolutions in attention models: setting, placement, initialization, and reported perplexity change (relative reduction where computable). "Width one": whether a no-mixing control was run.

study	setting	placement	init	reported change	width one
Canon (Allen-Zhu, 2025)	synthetic tasks and LM pretraining, dense and linear attention	Canon-A: $h + \mathrm{Conv}_4(h)$ on the normalized attention input	Kaiming	larger gains under linear than dense attention	no
Samba (Ren et al., 2025)	438M, 20B tokens, SWA (2048 window, 4096 context)	SiLU conv on Q, K, V	—	$11.12 \to 10.83$ (2.6%), throughput -5%; negative when added to both hybrid mixers	no
ConvLLM (Tian et al., 2026)	Qwen3-1.7B architecture from scratch, one run per site	pre-attention residual branch; concatenated Q/K/V outputs	varied	$13.42 \to 13.27$ (1.1%); $13.42 \to 12.85$ (4.2%)	no
Dynamic short conv. (Sieberling et al., 2026)	300M, 15B tokens, 4096 context	input-dependent conv on Q, K, V	static-equivalent	$19.12 \to 18.10$ (5.3%) at width four; 18.42 (3.7%) at width one	yes
This work	81M, 1.6B tokens, PG-19, dense and SWA (W=256, 8192 context)	residual stream before attention, own RMSNorm	zero taps	dense 0.59%; windowed 2.08%; width one 80–86% of windowed gain	yes

Table 20: Per-class paired NLL reduction under dense attention, mean $\pm$ SD over seeds 42/43/44, with each class's share of tokens and of the total loss reduction.

Context (tokens)	class	token share	Δ_{NLL} (mean $\pm$ SD)	% of total reduction
1024	novel bigram	79.4%	$\mathbf{+0.0086 \pm 0.0012}$	$88 \pm 5\%$
	distant recurrence	3.7%	$+0.0101 \pm 0.0068$	$5 \pm 4\%$
	local recurrence	16.9%	$+0.0032 \pm 0.0017$	$7 \pm 3\%$
4096	novel bigram	72.3%	$\mathbf{+0.0078 \pm 0.0024}$	$81 \pm 17\%$
	distant recurrence	12.0%	$+0.0080 \pm 0.0125$	$11 \pm 14\%$
	local recurrence	15.7%	$+0.0045 \pm 0.0039$	$8 \pm 5\%$
8192	novel bigram	64.9%	$\mathbf{+0.0082 \pm 0.0014}$	$101 \pm 36\%$
	distant recurrence	16.8%	$+0.0022 \pm 0.0059$	$4 \pm 19\%$
	local recurrence	18.3%	-0.0009 ± 0.0048	$-5 \pm 17\%$

vector, rescaled to the same norm and averaged over three draws, pre-registering the comparison. At 8192 tokens, proportional rescaling gives windowed/dense curvature-estimate ratios of 0.63–0.69 across seeds. Norm-matched nonradial perturbations give comparable curvature, with ratios of 0.99–1.04. The difference between ratios is 0.36 ± 0.03, consistent across all three seeds.

The first preregistered control addresses changes in DC gain, the sum of a channel's taps and thus its response to constant input. Random nonradial perturbations change this gain $1.21\times$ as much as radial perturbations. We therefore separate DC-aligned and zero-sum directions using eight draws and per-channel Gram–Schmidt orthogonalization. The anisotropy persists: at 8192 tokens, the zero-sum and DC ratios are 0.98 and 0.93, compared with 0.66 for radial rescaling. The difference therefore persists beyond DC changes. The DC direction is the most curvature-sensitive in both configurations, with 4–$6\times$ the radial curvature, but its curvature does not differ between configurations.

A second control reverses each channel's taps, projects the change orthogonally to the original tap vector, and matches its norm. Across four perturbation magnitudes, the resulting curvature is comparable between configurations, with ratios of 0.95–0.98.

Having checked coefficient directions, we next ask which layers contribute the contrast. When only layer 0's branch is scaled, the configurations are nearly equal (0.91). Scaling layers 1–9 alone reproduces the whole gap (0.64, against 0.66 for the whole branch). The windowed model thus has lower local curvature under fractional branch scaling (a robust $\sim 1/3$ reduction), while sensitivity to the tested coefficient-shape directions is equal across configurations. Necessity and finite removal are different quantities (above).

Perturbation-size and direction controls

Equal fractional rescaling need not move the residual representation by equal amounts in the two models. We therefore match displacement before comparing responses. To separate perturbation size, direction, and output metric, we inject δ at one site: the residual leaving the branch, feeding both attention normalization and the bypass. We measure $\mathrm{KL}(p_0 \| p_{+\delta}) + \mathrm{KL}(p_0 \| p_{-\delta})$. Here, p_0

Table 21: PG-19 paired NLL reductions under dense and windowed attention at 1024 and 4096 tokens (24 windows, mean ± SD across matched seeds), the shorter-context rows of Table 5. Dashes: unavailable.

Context (tokens)	class	dense	windowed	ratio	I_{NLL}
1024	novel bigram	$+0.0086 \pm 0.0012$	$+0.0108 \pm 0.0010$	$1.3\times$	—
	distant recurrence	$+0.0101 \pm 0.0068$	$+0.0015 \pm 0.0081$	—	—
	local recurrence	$+0.0032 \pm 0.0017$	$+0.0036 \pm 0.0008$	$1.1\times$	—
4096	novel bigram	$+0.0078 \pm 0.0024$	$+0.0153 \pm 0.0022$	$1.9\times$	—
	distant recurrence	$+0.0080 \pm 0.0125$	$+0.0091 \pm 0.0059$	—	—
	local recurrence	$+0.0045 \pm 0.0039$	$+0.0131 \pm 0.0027$	$2.9\times$	—

Table 22: Token-level decomposition by long-context sensitivity at an 8192-token context on PG-19, $K{=}256$, cross-fitted evaluator, nats per token, mean ± SD over seeds 42/43/44. Penalty $P_{\mathrm{base,NLL}}$ and interaction I_{NLL} are in nats/token; convolution gains are $\mathrm{NLL}_{\mathrm{base}} - \mathrm{NLL}_{\mathrm{conv}}$. Contribution is token share × mean, in 10^{-3} nats/token; the five bins plus the unclassified first 512 positions of each window sum to the corpus totals (penalty 14.4, interaction 14.9).

			convolution gain			contribution	
LSD bin	Token share (%)	penalty	dense	windowed	interaction	penalty	Interaction
≤ 0	39.7	-0.001 ± 0.002	$+0.006 \pm 0.002$	$+0.018 \pm 0.000$	$+0.013 \pm 0.001$	-0.4	$+5.0$
0–0.5	48.6	$+0.012 \pm 0.001$	$+0.006 \pm 0.001$	$+0.018 \pm 0.001$	$+0.011 \pm 0.002$	$+5.6$	$+5.4$
0.5–1	3.2	$+0.089 \pm 0.018$	$+0.007 \pm 0.005$	$+0.046 \pm 0.008$	$+0.039 \pm 0.002$	$+2.9$	$+1.3$
1–2	1.2	$+0.206 \pm 0.043$	-0.002 ± 0.017	$+0.074 \pm 0.007$	$+0.077 \pm 0.019$	$+2.5$	$+0.9$
> 2	1.0	$+0.368 \pm 0.066$	-0.020 ± 0.054	$+0.128 \pm 0.011$	$+0.148 \pm 0.064$	$+3.7$	$+1.5$
prefix (unclassified)	6.25	$+0.001 \pm 0.001$	$+0.009 \pm 0.002$	$+0.021 \pm 0.001$	$+0.012 \pm 0.002$	$+0.1$	$+0.7$

is the intact next-token distribution, and $p_{\pm\delta}$ are the distributions after the two perturbations. Displacement is matched before attention normalization at $\mathrm{RMS}(\delta)/\mathrm{RMS}(u) = 1\%$. Responses at 2% are within 4% of fourfold, consistent with approximately quadratic scaling.

We test branch-output perturbations at fixed fractional scaling (above) and matched displacement, together with output under a fixed channel permutation. Other directions are rank-16 slow activation-aligned perturbations (the trailing 256-token residual mean projected onto 16 principal components fitted on separate windows), random rank-16 coherent perturbations, and white noise. The coherent direction projects the same trailing residual mean onto a random 16-dimensional subspace.

Table 24 gives windowed/dense ratios (three seeds, 32 windows each).

Matching displacement changes the site-level comparisons. It brings the early-site branch ratios close to the white-noise ratios, leaves a branch-specific gap at site 4 that channel permutation substantially reduces, and shows no suppression of the slow direction.

To probe the remaining contrast, we vary the output metric and normalization. Ratios under the partner model's distribution and under the mean of the two differ from those under each model's own distribution by at most 0.03, so the metric does not explain the gap. Replaying the intact normalization denominator (normalizing the perturbed residual with the RMS denominator cached from the unperturbed pass) at site 4 is inconclusive (it removes the branch-specific gap in two seeds and not the third).

Two baseline controls probe the early-site white-noise asymmetry. Swapping downstream attention masks at inference shifts sensitivity in the opposite direction in every seed at both sites: windowed weights under a dense suffix become less sensitive, and dense weights under a windowed suffix become more sensitive. This rejects the immediate-mask explanation. Temporally correlated isotropic noise gives the same ratios as white noise (0.63 versus 0.63 at site 1), showing that temporal coherence alone is insufficient. The slow activation-aligned direction escapes the asymmetry in a way consistent with activation alignment rather than slowness.

Matching branch-output displacement raises the windowed/dense ratio of summed single-site responses from 0.57 to 0.79 across three seeds and sites 1, 2, 4, and 8 (the amplitude test above; Table 24). At the early sites dominating the response, windowed branches are smaller relative to

Table 23: Amplitude sensitivity across training context lengths (left) and inference-time attention-span restrictions in dense models (right). Left: amplitude-sensitivity statistic $C(0.05)$, mean $\pm$ SD across seeds, on the first 128 windows at each context length. Right: dense models trained at 8192 tokens, evaluated on a separate 96-window panel with every layer's attention span restricted; seed 42 shown, with three-seed ratios in the text. The windowed and span-restriction predictions fail; the dense clause is met by a flat result.

Context (tokens)	dense	windowed	windowed/dense
1024	0.00336 ± 0.00020	0.00344 ± 0.00069	$1.03\times$
4096	0.00308 ± 0.00021	0.00252 ± 0.00022	$0.82\times$
8192	0.00341 ± 0.00027	0.00218 ± 0.00016	$\mathbf{0.64}\times$

span (dense, 8192)	$C(0.05)$
full	0.003268
4096	0.003207
2048	0.003191
1024	0.003054
512	0.002963
256	**0.002895**

Table 24: Windowed/dense ratio of the two-sided predictive response to a single-site injection at 1% matched displacement (mean $\pm$ SD over seeds 42–44). "Fractional" is the branch scaled by 1 ± 0.05 as in the amplitude test above. Base-model rows have no branch; dashes denote unavailable entries.

models	site	fractional	matched	permuted	slow	coherent rnd.	white
+conv	1	0.54 ± 0.07	0.77 ± 0.07	0.95 ± 0.29	1.20 ± 0.17	0.98 ± 0.16	0.83 ± 0.16
+conv	2	0.71 ± 0.22	0.93 ± 0.12	0.86 ± 0.14	0.99 ± 0.07	1.10 ± 0.08	0.87 ± 0.12
+conv	4	0.65 ± 0.08	0.63 ± 0.08	0.90 ± 0.07	1.12 ± 0.22	0.98 ± 0.26	0.91 ± 0.01
+conv	8	0.98 ± 0.08	0.88 ± 0.11	0.97 ± 0.05	1.32 ± 0.15	0.89 ± 0.03	1.02 ± 0.08
base	1	—	—	—	1.09 ± 0.39	0.81 ± 0.10	0.63 ± 0.07
base	2	—	—	—	1.22 ± 0.11	1.03 ± 0.02	0.72 ± 0.08
base	4	—	—	—	1.12 ± 0.27	0.93 ± 0.02	0.90 ± 0.06
base	8	—	—	—	1.32 ± 0.14	0.92 ± 0.08	0.99 ± 0.04

the residual. The remaining asymmetry resembles that from white noise. Models trained without convolution show it more strongly, while changing only the attention mask shifts it the other way. Slow activation-aligned directions show no suppression; these project the trailing 256-token residual mean onto its leading 16 principal components fitted on separate windows. Thus the amplitude result reflects unequal activation displacements and endpoint sensitivity acquired under windowed training, not the training-time benefit, whose mechanism remains unexplained.

Attention statistics on matched inputs

A separate descriptive question is whether adding the branch changes where attention is directed. These statistics characterize that change without resolving the mechanism left open by the interventions. Using identical validation batches, we extract complete attention distributions from paired 1024-token checkpoints for two seeds. We measure per-layer previous-token attention mass, local mass at offsets 0–3 (the current token and the three preceding tokens), and support-normalized entropy. The last quantity divides attention entropy by the logarithm of the number of available key positions.

Both pairs show lower total previous-token attention mass with the convolution (-4.6% at seed 42, -2.7% at seed 43), mainly in middle and late layers. Layers 0–1 attend more locally. Normalized entropy is unchanged or slightly lower, suggesting that attention shifts its targets rather than becoming more diffuse.

These are correlational observations on matched inputs, consistent with a local-offloading account and inconsistent with a naive "freed capacity = broader attention" picture.

These interventions falsify specific accounts; they do not identify a mechanism.

F RUN PROVENANCE AND REPEATABILITY

Table 25 lists artifacts departing from "one run, tracker-complete, checkpoint intact." Three of the twelve PG-19 results at an 8192-token context († in Table 7) come from training logs. These runs completed 25,000 steps and attempted to save their final checkpoints. A CUDA error during NCCL teardown prevented the tracker's final synchronization, leaving histories at step 24,000. Where both records exist, the log and tracker agree exactly at every shared step. The collection script requires exact step matching, preventing comparisons between 24,000-step and 25,000-step results.

The seed-43 windowed baseline and seed-44 windowed convolution checkpoints are intact and were rescored at step 25,000 (Appendix B). The original seed-42 windowed baseline checkpoint was truncated by the crash and is unavailable for analysis. We reran that pair as `s42rr`; its result differs from the logged value by 0.11%. All checkpoint analyses, deterministic rescoring, and Figure 4 use the rerun pair. Table 7 retains the original logged result. The seed-42 windowed rows in the token-class and long-context-sensitivity decompositions also use this rerun pair.

Three runs were resumed from their last intact checkpoints and completed: two final checkpoints were truncated by the pre-fix teardown, and the seed-44 $W{=}1024$ convolution run on V100 stalled in the filesystem during its final save. The final 5,000 steps restored model and optimizer state but used fresh data sampling because the loop reseeds on resume. These continuations are equivalent in distribution, not bitwise reproductions, and are compared only after deterministic rescoring. They are exceptions to strict paired-seed attribution (P2 does not hold for their final steps), and their contrasts are treated as descriptive.

All A100 runs use bf16; runs marked V100 use fp16 with loss scaling; every deterministic rescoring and probe runs on V100/fp16; continuations run on the stack of their parent checkpoint (Table 25).

F.1 REPEATABILITY AND MEASUREMENT SCOPE

Accidental repeat runs of five configurations differed by 0.09–0.32%, providing a repeatability diagnostic rather than a calibrated floor. A deliberate duplicate of the seed-43 dense pair on one training stack agreed with the original baseline within 0.002% and with the original convolution configuration within 0.23%; the spread sits in the configuration that grows a branch from zero. Training the seed-43 dense pair on A100/bf16 and V100/fp16 changes each endpoint by 0.3% and its paired delta by $0.58\,\mathrm{pp}$ (Table 9). We therefore report uncertainty across seeds for each contrast rather than a fixed resolution threshold, compare deltas within one training stack, and treat single-pair differences of this size as descriptive. The paired runs execute as separate jobs, and floating-point reductions are not reproducible across machines.

The auxiliary window-sweep, candidate, budget, and transplant experiments are single-seed except where replication is stated; the scale extension has one pair at 253M. Mechanistic controls falsify specific accounts without identifying the interaction's cause.

G PRE-REGISTRATION CHRONOLOGY

Predictions were written into the project's version-controlled experiment log with their decision rules; Table 26 gives each original prediction, its registration and launch dates (from the commit history, local time, 2026), what had been observed when it was written, and its outcome under the original rule. Rows 2–4 and 6–8 were registered before any test result existed; rows 1 and 9 were registered after their test runs had launched but before they completed; row 5 was registered at launch. Registered text was only appended to, with one exception: row 5's reference dense endpoints were corrected in place in the commit that logged its first result. Where we later changed an analysis (deterministic rescoring, scoping a failed test), the table says so. Commit references are withheld for anonymity and will be released with the camera-ready version.

Table 25: Provenance of results requiring recovered logs, reruns, or resumed training.

artifact	provenance
8192 seed-42 windowed base	original completed; value from log (18.89); checkpoint truncated. Rerun `s42rr` (18.87) used for all checkpoint work and Figure 4.
8192 seed-42 windowed convolution	original tracker-complete; a rerun partner exists and is used with the base rerun for checkpoint work.
8192 seed-43 windowed base; seed-44 windowed convolution	completed; tracker ends at 24,000; step-25,000 value from log; checkpoints intact and rescored (Appendix B).
$1.8\times$ windowed convolution (45k)	final checkpoint truncated; steps 40k–45k resumed and completed from the 40k checkpoint (optimizer restored, fresh sampling); rescored alongside the comparison configurations on the same training stack (§ 5.3).
Transplant, dense taps frozen	final checkpoint truncated; steps 20k–25k resumed and completed from the 20k checkpoint as above; rescored alongside the comparison configurations on the same training stack (§ 5.4).
$W{=}1024$ convolution, seed 44 (V100)	final save hung in a CephFS stall after the step-25k validation; steps 20k–25k resumed and completed from the 20k checkpoint as above; rescored alongside the comparison configurations on the same training stack (Table 10).
$W{=}64/512$ seeds 43/44; hybrid factorial; initialization control; continuations; perturbation panels	trained or run on V100/fp16; contrasts within stack only (Table 9).
Step-15k branch insertion/removal and shams	A100 continuations from the A100 step-15k checkpoints of seed 42; rescored deterministically with their endpoints (§ 5.4).
Operator family (bilinear, dynamic-tap, sigmoid-gated), seeds 42–44, and the seed-42 dense convolution on V100	trained on V100/fp16 and rescored deterministically against same-stack baselines (Table 18); the seed-42 dense bilinear run hung in its step-20k save, so steps 15k–25k were resumed from the 15k checkpoint as above.
Pointwise control, bilinear-operator pilot, duplicate pair (seed 43)	trained on V100/fp16 in one batch and rescored deterministically with the V100 dense pair; seed-42 pointwise configuration trained on A100 and rescored with the seed-42 A100 windowed pair; seed-44 triple trained on V100/fp16 and rescored deterministically.

Table 26: Preregistered predictions and their outcomes under the original rules, in order of registration. Reg./launch: registration and test-launch dates. "Seen": what was already observed at registration.

#	original prediction and rule	reg. / launch	seen	outcome under the rule
1	Interaction tracks the window penalty about one-for-one: at 4096, interaction $\approx +0.9$ pp and windowed+conv ≈ 19.26; at 1024, windowed+conv ≈ 19.60 (point predictions, seed 42)	08-18 / 08-17	8192 seed 42 (discovery); three of four cells of each test grid; test runs in progress	missed at both: 4096 19.39 ($+0.19$ pp), 1024 19.72 ($+0.24$ pp), training-time; three-seed rescoring later gives $+0.52 \pm 0.37$ pp and $+0.11 \pm 0.08$ pp; 8192 replicated at seeds 43/44
2	Amplitude sensitivity C rises with context under windowing and falls or stays flat under dense attention; restricting a dense model's span raises it. Falsified if dense C rises as fast as windowed C, or no arm$\times$context interaction exceeds seed spread; then "withdrawn, not reworded"	08-21 / 08-21	endpoint perplexities; an earlier full-removal probe	falsified (Table 23); hypothesis withdrawn as registered. Limiting its scope to amplitude (Appendix E) is our later reading
3	At 8192, nonradial windowed/dense curvature ratio ≥ 1.0, above the radial ratio, with larger nonradial/radial anisotropy in the windowed model; falsified if within ± 0.1 of the radial ratio or below it	08-22 / 08-22	radial ratio 0.76 (norm-matched; 0.64 raw)	falsifier not triggered and anisotropy clause met; the ≥ 1.0 clause is unresolved: $1.042/1.011/0.993$, mean 1.015, 95% interval $[0.955, 1.076]$
4	Transplanted own taps within 0.06 perplexity of zero initialization = equivalent; ≥ 0.10 = material; between = inconclusive (training-time validation)	08-22 / 08-22	none	training-time gap $+0.10$, at the materiality margin; deterministic rescoring, not registered, gives $+0.083$, between the margins (§ 5.4)
5	At 8192, interaction monotone in penalty across windows: larger at $W{=}64/128$ than 256, smaller at 512/1024, $I(1024)$ closest to zero; falsified if non-monotone or $I(64) \leq I(1024)$	08-28 / 08-28	$W{=}256$ and dense pairs; $r{=}0.82$ across contexts	falsified: $I(64) = -0.41$ pp $\leq I(1024) = +0.84$ pp (Table 8); the penalty itself peaks at $W{=}256$. The tracking reported in § 5.3 is post hoc description
6	At $K{=}256$, per-token window penalty rises monotonically with LSD bin, and bins with LSD > 1 carry a majority of the penalty	08-28 / 08-28	no binned data	monotone clause met ($32\times$ across bins); majority clause narrowly fails (43%; Table 22)
7	If cross-layer cancellation explains the curvature gap: independent per-layer scaling ratio in $[0.9, 1.1]$, and cross-layer terms carry $\geq$ 50–70% of the gap	09-09 / 09-09	joint ratio 0.64	both clauses fail: independent ratio 0.55–0.65; cross-layer terms $\approx$57% of the response in both models, so they carry none of the gap (Appendix E)
8	Gradual withdrawal retains $\geq 70\%$ of the sham interaction if the benefit is stored in shared weights	09-09 / 09-09	none	retained $+0.017$ versus 0.7×0.015: nominally passes, only because the dense advantage is small; neither withdrawn model matches its baseline (costs $0.023/0.025$ nats)
9	At 1% matched displacement: native ratio ≥ 0.90 with fractional ratio $\approx$0.6 $\Rightarrow$ perturbation size; native and structured ≤ 0.75 with random directions 0.90–1.10 $\Rightarrow$ transmission; gap under own but not partner distribution $\Rightarrow$ output metric; between bands $\Rightarrow$ mixed	09-11 / 09-11 (3 min earlier)	fractional ratio $\approx$0.6; runs launched	native per site $0.77/0.93/0.63/0.88$: between bands for size; structured ≥ 1, reversed, so transmission rejected; ratios change ≤ 0.03 across distributions, so output metric rejected (Table 24)