ON THE EFFECTIVENESS AND INDUCTIVE BIAS OF BLOCK GROUPING IN ATTENTION RESIDUALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Attention Residuals (AttnRes) make residual history addressable, raising a natural follow-up question: what should constitute a unit of retrieval? Full AttnRes exposes every sublayer output independently, whereas Block AttnRes groups outputs and forces them to share a readout coefficient. We find that this restriction can improve both learning and efficiency. On FineWeb-Edu, Block outperforms Full for all three seeds at 124M parameters and both seeds at 350M under our standard recipe, reversing the original AttnRes ordering. To separate candidate count from candidate construction, we compare Block with a local non-aligned control matching its final group count and sizes, with similar per-reader candidate counts. Block remains better across the reported comparisons, showing that candidate-count reduction alone is insufficient to explain the gain. Learning-rate sweeps at both scales and a fourfold training-budget range at 124M show that the Block–Full ordering survives retuning, even when retuning restores Full's advantage over standard residuals. With a fused Triton implementation, Block uses approximately 14% and 26% less GPU time per token than Full at 124M and 350M. These results show that residual-source grouping is not merely an efficiency choice but a meaningful inductive bias: how residual history is partitioned can matter as much as how finely it is accessed.

1 INTRODUCTION

Residual connections determine how information produced at different depths remains available to later computation. In a standard Transformer, this access pattern is fixed: attention and MLP outputs are accumulated into a single residual stream, and later layers consume only their sum. (He et al., 2016; Vaswani et al., 2017) Attention Residuals (AttnRes) replace this fixed accumulation with learned depth-wise readout, allowing later layers to selectively retrieve earlier outputs (Kimi Team et al., 2026). Once residual history becomes addressable, however, a new architectural question arises: *what should constitute an addressable unit of that history?*

Full AttnRes makes every sublayer output an independent candidate, maximizing the granularity of depth-wise access. Block AttnRes instead sums multiple outputs before they are exposed to later readers, forcing the members of each group to share a readout coefficient. This restriction clearly reduces computation, but it also changes the hypothesis class of the residual pathway: once two outputs are grouped, no later reader can reweight them independently. Grouping therefore does not merely compress the candidate set; it defines the basis over which depth-wise attention operates. Whether finer-grained access is actually preferable is consequently a question about the inductive bias of residual routing, not only its efficiency.

The answer is not obvious empirically. The original AttnRes study reports lower loss for Full and identifies finer grouping as a promising direction (Kimi Team et al., 2026), whereas subsequent work reports a Block configuration outperforming Full (Su, 2026). This leaves two interesting possibilities: Block may simply be a cheaper approximation whose apparent advantage depends on scale or optimization, or the structural restriction imposed by grouping may itself improve learning. We investigate this distinction by comparing Full and Block across model scales, training recipes, and grouping controls designed to separate candidate count from candidate construction.

On FineWeb-Edu (Penedo et al., 2024), we find a consistent advantage for Block. Under our standard recipe, Block achieves lower validation loss than Full for all three seeds at 124M parameters and

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.

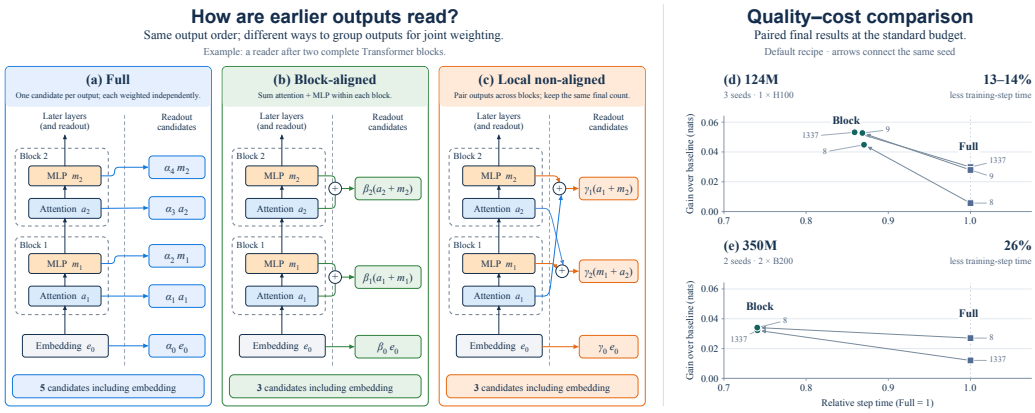


Figure 1: **Readout grouping and quality–cost comparison:** Full reads individual outputs, while grouped outputs share a readout coefficient. Default-recipe results cover three seeds at 124M and two at 350M, plotting same-seed baseline-minus-arm validation loss against median training-step time normalized to the same-seed Full run.

both seeds at 350M. The advantage does not come with an efficiency tradeoff: using a shared fused implementation, Block also requires less measured training time per token than Full at both scales.

We next ask why grouping helps. A natural explanation is that Block benefits simply from reducing the number of candidates presented to each reader. To test this, we compare block-aligned summaries with non-aligned groupings at closely matched candidate budgets. Block outperforms the controls across the reported comparisons, while summaries spanning two complete Transformer blocks nearly match one-block summaries in our 124M granularity sweep. Together, these results indicate that candidate count alone is insufficient to explain Block's advantage and instead favor a structural prior that preserves complete Transformer blocks within residual summaries.

Architectural comparisons can also be confounded by optimization. (Melis et al., 2018; Bello et al., 2021) At 124M, Full's gain over standard residuals nearly disappears as training is extended under the default learning rate. After retuning, that gain reappears, showing that the apparent budget trend is recipe-dependent. The ordering between Block and Full, however, remains unchanged when each architecture is evaluated at its best tested learning rate, and learning-rate sweeps at 350M show the same ordering across the stable range. This distinction matters for evaluating residual architectures: fixed-recipe scaling can obscure whether a gain is architectural or merely optimization-dependent.

To summarize, our contributions are threefold:

- **Finer-grained residual access is not necessarily better.** Across our 100+ run experimental campaign, Block consistently outperforms the more fine-grained Full AttnRes in the paired regimes we study while also reducing measured training cost. This challenges the interpretation of Block as merely an efficiency-oriented approximation to Full.

- **The candidate basis of depth-wise attention is itself an inductive bias.** To our knowledge, this is the first study to test Transformer-block alignment in AttnRes against shifted and local non-aligned grouping controls. Our grouping controls show that candidate-count reduction alone is insufficient to explain Block's advantage and favor summaries that preserve complete Transformer blocks. More broadly, they identify residual-source construction—not only learned readout weights—as a meaningful architectural design axis for depth-wise attention.

- **Architectural scaling conclusions require optimization-aware evaluation.** Retuning changes the apparent persistence of Full's gain over standard residuals without reversing the Block–Full ordering. This concrete case study shows how fixed-recipe trends can be mistaken for architectural limits, motivating budget-specific tuning before drawing conclusions about longer training.

2 Related Work

Residual Connections & Learned Weights Residual connections ease optimization by supporting signal propagation in deep networks, becoming foundational to vision and language architectures. (He et al., 2016; Vaswani et al., 2017; Devlin et al., 2019; Dosovitskiy, 2020) Early approaches explored learned connectivity through Highway gates and weighted residual branches. (Srivastava et al., 2015; Shen et al., 2016) DLCL (Wang et al., 2019) extended weighting to combinations of preceding layer representations. Further efforts such as ReZero (Bachlechner et al., 2021) and LayerScale (Touvron et al., 2021) introduced learnable scalar and channel-wise residual scaling, respectively, with initialization designed to facilitate optimization. By contrast, DeepNet (Wang et al., 2024) stabilized deep Transformers through theoretically derived, fixed residual scaling and tailored initialization. More recently, works such as Hyper-Connections Zhu et al. (2025) and mHC (Xie et al., 2025) generalized residual pathways to learnable mixing across multiple streams. Taken together, these methods control residual information flow through complementary mechanisms, from branch scaling to learned multi-stream aggregation.

Attention & Attention Residuals Attention enables adaptive information selection in translation, image captioning, and Transformers. (Bahdanau et al., 2014; Luong et al., 2015; Xu et al., 2015; Vaswani et al., 2017) Following cross-layer aggregation in DenseFormer (Pagliardini et al., 2024), ANCRe (Zhang et al., 2026) and DeepCrossAttention (Heddes et al., 2025), Attention Residuals (Kimi Team et al., 2026) extends this principle across network depth, replacing fixed residual accumulation with input-dependent attention over earlier layer outputs and improving language modeling and downstream performance. Its Block variant reduces memory and communication costs while preserving most of Full AttnRes's gains. (Kimi Team et al., 2026) However, given near-parity in some original scaling comparisons and mixed subsequent findings, it is not immediately clear whether Block is consistently inferior to Full. (Kimi Team et al., 2026; Su, 2026) We systematically investigate their relative effectiveness across model scales, training budgets, learning rates, and random seeds. Per the importance of appropriate structural priors (Battaglia et al., 2018; d'Ascoli et al., 2021), we additionally investigate the inductive bias of Transformer-block-aligned residual aggregation.

3 Background: Attention Residuals and Grouped Readout

Attention Residuals. A pre-norm Transformer (Vaswani et al., 2017; Xiong et al., 2020) with L blocks produces $2L$ sublayer outputs, the attention output $\boldsymbol{a}_i$ and the MLP output $\boldsymbol{m}_i$ of block i, plus the token embedding $\boldsymbol{e}_0$. In the standard architecture (He et al., 2016) the input to every sublayer is the running sum of the embedding and all sublayer outputs produced so far; Attention Residuals (Kimi Team et al., 2026) replace this fixed sum with learned, input-dependent attention over the individual outputs. Index the sources in production order, $s_0 = \boldsymbol{e}_0$, $s_1 = \boldsymbol{a}_1$, $s_2 = \boldsymbol{m}_1$, …, $s_{2L} = \boldsymbol{m}_L$. There are $2L{+}1$ readers: reader k is the input to the $(k{+}1)$-th sublayer for $k < 2L$ and the final readout before the output norm for $k = 2L$; it attends over the sources available to it,

$$\boldsymbol{h}_k \;=\; \sum_{j\le k} \alpha_{k,j}\, s_j, \qquad \alpha_{k,\cdot} \;=\; \mathrm{softmax}_{j\le k}\big(\boldsymbol{q}_k^{\top}(\boldsymbol{g}_k \odot \hat{s}_j)\big), \qquad \hat{s}_j \;=\; \frac{s_j}{\sqrt{d^{-1}\|s_j\|_2^2 + \epsilon}}, \quad (1)$$

with a learned query $\boldsymbol{q}_k \in \mathbb{R}^d$ per reader and RMS-normalized keys (Zhang & Sennrich, 2019) with a learned gain $\boldsymbol{g}_k$ per reader; algebraically the logits depend only on the product $\boldsymbol{q}_k \odot \boldsymbol{g}_k$. The values are the raw sources. We call this formulation *Full*: every source, the embedding and each earlier sublayer output, is its own candidate, $2L{+}1$ of them at the final reader. The queries are zero-initialized, so at initialization every reader returns the uniform average of its sources, the standard residual sum scaled by $1/(k{+}1)$. The normalization that consumes $\boldsymbol{h}_k$ removes this scalar exactly only if its ϵ is zero, since $\mathrm{RMSNorm}_\epsilon(\boldsymbol{x}/c) = \boldsymbol{x}/\sqrt{d^{-1}\|\boldsymbol{x}\|_2^2 + c^2\epsilon}$; at the default ϵ the scaling acts as an effective epsilon of $c^2\epsilon$ (Appendix A.4).

Grouped readout. A *grouping* partitions the source indices into buckets $B_1, \ldots, B_G$. Reader k can address a bucket once one of its members exists, as the running sum of its members so far,

$$c_{k,g} \;=\; \sum_{j\in B_g,\; j\le k} s_j, \quad (2)$$

so the members of a bucket are summed before scoring and *share one readout coefficient* $\alpha_{k,g}$. The reader applies Eq. (1) to its active candidates, with the RMS-normalized candidate as key and the candidate itself as value. *Block* AttnRes (Kimi Team et al., 2026) partitions the sublayers into consecutive blocks of chosen size, keeps standard accumulation inside a block and attends over completed block sums, the current partial block and the embedding. Our Block configuration, one Transformer block per bucket, is the finest block size keeping attention and MLP outputs together,

$$\{\boldsymbol{e}_0\},\ \{\boldsymbol{a}_1, \boldsymbol{m}_1\},\ \{\boldsymbol{a}_2, \boldsymbol{m}_2\},\ \ldots,\ \{\boldsymbol{a}_L, \boldsymbol{m}_L\}, \tag{3}$$

giving $L{+}1$ candidates at the final reader; coarser block sizes appear in the granularity sweep of Section 4.3. Since the active candidates of any partition sum to the same stream, zero queries make reader k return that stream scaled by $1/G_k$, the inverse number of its active candidates ($G_k = k{+}1$ for Full, $\lceil k/2 \rceil{+}1$ for Block): all groupings recover the standard-residual function exactly at $\epsilon = 0$ and differ slightly at finite ϵ because G_k differs. They differ further in how earlier outputs are combined into candidates before scoring, and hence in what each reader can address.

What a reader actually sees. Because candidates are running sums, a bucket is *partial* for every reader that runs before its last member exists: under Eq. (3) the MLP of block i reads $\boldsymbol{a}_i$ alone, later readers $\boldsymbol{a}_i + \boldsymbol{m}_i$. Partitions with identical final bucket sizes can therefore expose different candidate and partial-bucket counts to intermediate readers; Section 4.3 reports both per reader for every grouping we compare and uses them to state how closely each comparison is matched.

4 Experimental Design

4.1 Shared Setup

Models and data. Decoder-only Transformers at two scales are trained on FineWeb-Edu (Penedo et al., 2024): 124M parameters (12 blocks, width 768) and 350M (24 blocks, width 1024), with sequence length 2,048 and a global batch of 256 sequences. The token stream is deterministic and seed-independent, so all runs on a data sample consume the same tokens per optimizer step; the seed sets only the initialization, which is identical across arms because the depth-attention parameters are initialized deterministically. The *standard* budgets are 4,800 steps (2.5B tokens) at 124M and 13,700 steps (7.2B tokens) at 350M, about 20 tokens per parameter; longer budgets appear in Section 4.4.

Shared training and implementation. All runs use AdamW (Loshchilov & Hutter, 2019) with a cosine schedule to 10% of the peak rate, 200 warm-up steps at 124M and 300 at 350M, weight decay 0.1 and gradient clipping at 1.0; the default peak rate is 6×10^{-4} at 124M and 3×10^{-4} at 350M. Every arm runs from one codebase, differing only in its residual mode, and every depth-attention variant uses the same fused Triton kernel (Tillet et al., 2019). The 124M runs use one H100; the 350M runs use B200s, one for the baseline and two for the depth-attention arms, with the global batch unchanged. Full configurations are in Supplementary Table S12.

Evaluation and pairing. The metric is held-out FineWeb-Edu cross-entropy at the predetermined end of training. For a variant and its baseline trained with the same seed, budget and recipe we report $\Delta = \mathcal{L}_{\text{baseline}} - \mathcal{L}_{\text{variant}}$, so positive values are improvements. Two FineWeb-Edu samples are used (Supplementary Table S13): sample A for the 124M main matrix, granularity sweep, epsilon control and default-recipe grouping controls; the larger sample B for the 124M budget and learning-rate campaign, including its default-rate runs, the tuned-recipe grouping controls and all 350M experiments. Every reported loss difference is within one sample. Seeds and partitions were fixed before the standardized campaign, and every standardized run is reported (Appendix E).

4.2 Quality and Cost Comparison

The first question is how Full and Block compare in quality and running cost under the standard recipe. The arms are the standard-residual baseline, Full and Block (Section 3); a static-skeleton variant is in Appendix A.1. Coverage is three seeds at 124M and two at 350M at the standard budgets (Table 2). Quality is endpoint validation loss, as Δ and as the paired Full–Block difference.

Figure 1 uses median training-step times from complete 10-update logging intervals after step 300. The logger excludes evaluation and checkpoint pauses. Full and Block use equal GPU counts within

Table 1: What each reader sees, 12-layer model. *Candidates per reader* is the mean number of addressable buckets over the $2L+1$ readers; *partial buckets* is the mean number of buckets with unproduced members.

Grouping	Final buckets	Cand. / reader	Partial / reader	Aligned pairs
Block (aligned)	13	7.24	0.48	12
Local non-aligned random	13	7.48	0.96	**0**
Half-block shifted	14	7.72	0.44	0
Global random pairs	13	**8.96**	**3.92**	0
Full	25	13.00	0	—

each scale, so their step-time and GPU-time-per-token ratios coincide at the fixed global batch. At 350M the baseline uses one card and depth-attention arms use two; the baseline-relative GPU-time-per-token ratios in Supplementary Table S2 account for this difference. These are training-only costs, excluding end-to-end job overhead; per-run medians are in Supplementary Table S3. A single-seed control at 124M reruns Full and Block with the RMSNorm epsilon set to zero, since the zero-initialized stream is divided by the pool size; its setup and result are in Appendix A.4.

4.3 Aggregation Granularity and Alignment

The second question is whether Block helps by having fewer candidates or by *which* outputs it merges. Keeping block boundaries, we vary how many consecutive complete blocks share one candidate: one (Block), two, three or four, with the embedding a singleton and Full as reference, in a single-seed sweep at 124M (seed 9, default recipe). The primary control is *local non-aligned* grouping: within each non-overlapping window of two consecutive blocks, $(1,2),(3,4),\ldots$, with outputs $\{\boldsymbol{a}_i, \boldsymbol{m}_i, \boldsymbol{a}_{i+1}, \boldsymbol{m}_{i+1}\}$, the outputs are paired by one of the two perfect matchings that avoid a block's own attention–MLP pair, type pairs $\{\boldsymbol{a}_i, \boldsymbol{a}_{i+1}\}, \{\boldsymbol{m}_i, \boldsymbol{m}_{i+1}\}$ or crossed pairs $\{\boldsymbol{a}_i, \boldsymbol{m}_{i+1}\}, \{\boldsymbol{m}_i, \boldsymbol{a}_{i+1}\}$, drawn per window; for each model scale one realization was fixed before any training run and reused at every training seed and recipe at that scale (Supplementary Table S14). The result has Block's final profile, $\{\boldsymbol{e}_0\}$ plus L pairs, and no aligned pair. The complementary *shifted* control keeps adjacent pairing but moves the boundary by one sublayer, pairing each block's MLP output with the next block's attention output, $\{\boldsymbol{m}_i, \boldsymbol{a}_{i+1}\}$; $\boldsymbol{e}_0$, $\boldsymbol{a}_1$ and $\boldsymbol{m}_L$ stay singletons, giving $L+2$ buckets to Block's $L+1$. A *global random* pairing of the $2L$ outputs is a supplementary check on depth locality, reported in the appendix.

Table 1 reports, for each grouping, the mean number of addressable candidates and partial buckets over the $2L+1$ readers at 12 layers. The local control matches Block's final bucket count and size profile and is close to Block on average (7.48 versus 7.24 candidates, 0.96 versus 0.48 partial buckets); the shifted control is similarly close with one extra bucket, whereas global random pairing exposes intermediate readers to markedly more candidates. The local control thus tests grouping structure at an exactly matched final bucket count and size profile, and the shifted control is a near-matched complement with one extra bucket; neither isolates alignment alone, because intermediate exposure is not identical. The analogous 24-layer audit is in Supplementary Table S6.

At 124M the local, shifted and global random controls were run with seeds 9 and 1337 at the default rate and at the common 2.4×10^{-3} rate selected by the main-arm sweep of Section 4.4, without individual retuning of the controls; at 350M the local and global random controls were run with seed 1337 under the default recipe. Other groupings at 124M (same-type bands, telescoping resolution, a profile-derived merge, and a shifted variant that folds the embedding into the first bucket), mostly single-seed, are in Appendix B.6. We also measure per-token norms and cosines of aligned, shifted and random pairs on trained 124M checkpoints, a descriptive check on the magnitude explanation whose protocol and table are in Appendix B.4.

4.4 Training-Budget and Learning-Rate Sensitivity

The third question is whether conclusions about architecture gains change with training budget and learning rate. Budgets at 124M are $1\times$, $2\times$ and $4\times$ the standard (4,800, 9,600 and 19,200 steps; 2.5B, 5B and 10B tokens), each a complete cosine schedule trained from scratch on the same deter-

Table 2: Default-recipe comparison. Validation loss in nats on the FineWeb-Edu holdout; Δ is the same-seed baseline minus the arm (positive is better) and the last column is Full minus Block (positive favours Block). The full table including the static-skeleton arm is Supplementary Table S1.

Scale	Seed	Baseline	Full	Δ Full	Block	Δ Block	Full $-$ Block
124M	1337	3.27251	3.24267	+0.0298	**3.21932**	**+0.0532**	+0.0234
	8	3.27788	3.27222	+0.0057	**3.23305**	**+0.0448**	+0.0392
	9	3.27742	3.24952	+0.0279	**3.22470**	**+0.0527**	+0.0248
	mean			+0.0211		**+0.0502**	+0.0291
350M	1337	2.88258	2.87055	+0.0120	**2.85036**	**+0.0322**	+0.0202
	8	2.87957	2.85256	+0.0270	**2.84549**	**+0.0341**	+0.0071
	mean			+0.0195		**+0.0332**	+0.0136

ministic token stream, so a longer run extends the tokens of a shorter one rather than resuming its checkpoint. The 350M experiments use the standard budget only.

Peak rates are swept over $\{1, 2, 4, 8\}\times$ the default at 124M and $\{1, 2, 4, 8, 16\}\times$ at 350M, each a complete schedule. At 124M the full grid was run at $1\times$ and $4\times$ on seed 9 and the $2\times$ budget covers only the two lowest rates; at 350M the five rates were run on seed 1337. The *best tested* rate is defined per arm and scale, at the budgets whose sweep brackets the optimum ($1\times$ and $4\times$ at 124M, the standard budget at 350M), as the grid rate with the lowest endpoint loss on the sweep seed, the baseline included; the $2\times$ runs are an intermediate-budget check without retuning; a second seed (1337 at 124M) reuses the rate selected on the sweep seed, and default-rate runs at $1\times$ and $4\times$ also exist for seeds 8 and 1337. Complete grids, additional seeds and trajectories past the best rate are in Appendix C.

5 Results and Discussion

5.1 Block Outperforms Full at Lower Training Cost

Block achieves lower validation loss than Full in all five paired comparisons under the standard recipe: three seeds at 124M and two at 350M (Table 2). Averaged over seeds, its gain over standard residuals is 0.0502 nats at 124M, compared with 0.0211 for Full, and 0.0332 nats at 350M, compared with 0.0195. With a shared fused implementation, Block's median training-step time is 13–14% lower than Full's at 124M and 26% lower at 350M (Figure 1). Because Full and Block use the same GPU count and global batch within each scale, these step-time reductions are the same percentage reductions in measured GPU time per token. Block therefore lowers validation loss while reducing measured GPU time per token in these settings. Full's finer readout lets later readers reweight individual sublayer outputs independently, but this additional readout freedom does not translate into lower validation loss here.

The ordering survives a check for pool-size-dependent normalization effects. At initialization, uniform readout averages different numbers of candidates in Full and Block, changing the effective epsilon in the subsequent RMSNorm. Setting epsilon to zero in a paired 124M run (seed 9) leaves a Block–Full gap of 0.0270 nats, compared with 0.0248 under the default epsilon. Block's advantage therefore persists after removing this specific source of numerical asymmetry.

Block therefore merits evaluation as an architectural choice in its own right. The original AttnRes experiments favored Full (Kimi Team et al., 2026), whereas a subsequent study reported lower loss for one Block configuration under a different training setup (Su, 2026). Here, Block's lower loss recurs across seeds at two scales and accompanies lower measured training cost. These experiments do not determine the ordering in the original study's architecture or training regime: our largest model is 350M at approximately 20 training tokens per parameter, and our longest runs reach approximately 80 tokens per parameter at 124M.

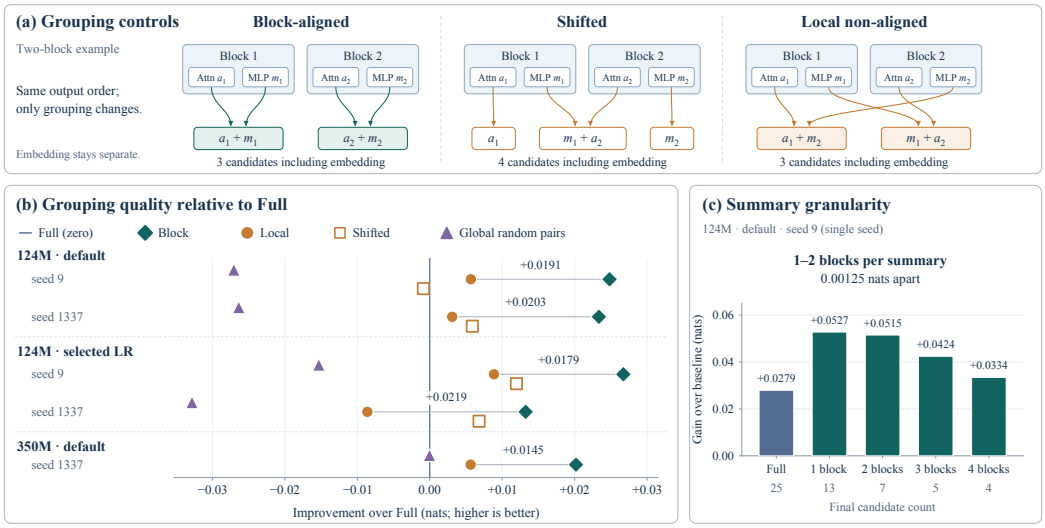


Figure 2: **Grouping structure and granularity:** (a) compares aligned, shifted, and local non-aligned candidates without changing execution order. (b) reports $L_{\text{Full}} - L_{\text{grouping}}$ within each condition; gray connectors label the paired advantage of Block over Local, $L_{\text{Local}} - L_{\text{Block}}$. (c) shows baseline-minus-arm gains for the 124M default seed-9 granularity sweep, where one- and two-block summaries differ by 0.00125 nats.

Table 3: Gain over the same-seed baseline (Δ, nats) for every grouping. 124M: default recipe (peak LR 6×10^{-4}) and tuned recipe (2.4×10^{-3}), seeds 9 and 1337; 350M: default recipe (3×10^{-4}), seed 1337. Baselines: 3.27742 / 3.27251 (124M default), 3.13761 / 3.12959 (124M tuned), 2.88258 (350M).

	124M, default		124M, tuned		350M
Grouping	s9	s1337	s9	s1337	s1337
Block (aligned)	**+0.0527**	**+0.0532**	**+0.0768**	**+0.0515**	**+0.0322**
Half-block shifted (`shift14`)	+0.0270	+0.0357	+0.0620	+0.0451	—
Local non-aligned random (`randloc`)	+0.0336	+0.0329	+0.0589	+0.0297	+0.0177
Full	+0.0279	+0.0298	+0.0501	+0.0383	+0.0120
Global random pairs (`rand13`)	+0.0009	+0.0035	+0.0348	+0.0055	+0.0120

5.2 Grouping Structure Matters Beyond Candidate Count

Matching Block's final candidate count without its block-aligned grouping does not reproduce Block's gain. The local non-aligned control has the same final candidate count and group sizes as Block, but pairs outputs across neighboring Transformer blocks. Block outperforms it in all five comparisons: by 0.0179–0.0219 nats across two seeds and two recipes at 124M, and by 0.0145 nats at 350M (Figure 2b; Table 3). The shifted control preserves adjacent pairing while moving the grouping boundary by one sublayer; Block also leads this control in all four 124M comparisons, by 0.0064–0.0257 nats. Both controls lie close to Full, within 0.009 nats for the local control and 0.012 for the shifted control. The combination of candidate reduction and depth-local grouping is therefore insufficient to reproduce Block's full advantage in these comparisons.

The granularity sweep suggests that this benefit does not require a separate candidate for every Transformer block. Summaries spanning two complete blocks nearly match single-block summaries, differing by only 0.00125 nats while reducing the final candidate count from 13 to 7 (Figure 2c; Supplementary Table S4). Summaries spanning three or four blocks degrade gradually but still achieve lower loss than Full. This single-seed 124M sweep supports preserving complete blocks within summaries, without establishing a uniquely preferred summary size.

Candidate statistics offer little support for a simple explanation in which aligned sums help because they are larger. On the trained Full model, aligned pairs are more correlated than random pairs, but the norms of their summed vectors are only about 3% larger than those of random pairs; on

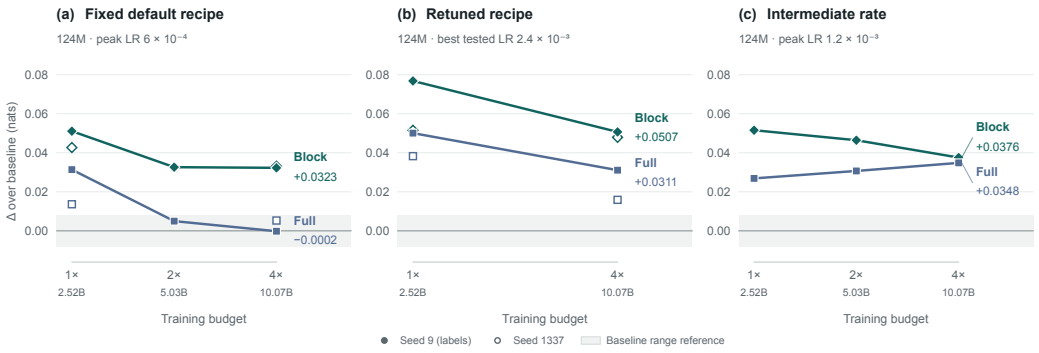


Figure 3: **Recipe-dependent budget trends:** Completed 124M cosine schedules report paired gain $\Delta = L_{\text{baseline}} - L_{\text{arm}}$ under (a) default, (b) retuned, and (c) intermediate (1.2×10^{-3}) learning rates. Filled markers and endpoint labels denote seed 9, open markers denote seed 1337; the intermediate-rate panel uses seed 9 only. The retuned $2\times$ budget was not measured; Full approaches parity only at the default rate, while Block remains ahead. The gray band spans ± 0.00802 nats, the difference between the two baseline seeds at the $1\times$ retuned setting, a descriptive scale reference rather than a confidence interval.

the trained Block model, aligned sums are no larger than hypothetical shifted sums (Supplementary Table S7). The global-random model has the highest correlation within its assigned pairs, yet the worst validation loss among these audited 124M models. These endpoint measurements are consistent with representations adapting to the imposed grouping, but such adaptation does not make arbitrary groupings equally effective. They also do not exclude magnitude effects: routing keys are RMS-normalized, while the values remain unnormalized.

Together, these results show that the candidate basis is not a neutral implementation detail of depth-wise attention. A grouping precommits which earlier outputs become inseparable once they are jointly available: subsequent readers must weight their sum with a single coefficient. Attention is therefore adaptive only over the candidate basis exposed by the architecture; learned coefficients do not remove this inductive bias. In our comparisons, the strongest groupings preserve complete Transformer blocks within summaries, while the granularity sweep points to boundary preservation rather than a uniquely optimal one-block summary size as the relevant structural prior. These comparisons do not isolate alignment alone: the local control exposes intermediate readers to slightly more candidates and more partial buckets, while the shifted control has one extra final candidate (Table 1). The local partition is fixed across training seeds, and the 350M grouping comparison covers one seed at the default recipe. The evidence supports block-aligned aggregation as a useful structural prior in these settings, while the mechanism behind its advantage remains open.

5.3 Budget Trends Change After Retuning

At a fixed default learning rate, longer training appears to erase Full's advantage over standard residuals. In the 124M budget campaign, Full's gain falls from 0.0314 nats at $1\times$ to 0.0050 at $2\times$ and -0.0002 at $4\times$ on seed 9, while Block retains a gain of 0.0323 at $4\times$ (Figure 3a). The three-seed means at $4\times$ show the same pattern (Supplementary Table S9): 0.0003 nats for Full and 0.0303 for Block. Taken alone, these results invite an explanation in which Full offers only a transient optimization benefit, whereas Block's advantage persists because of its aggregation structure. Retuning tests whether that interpretation survives a change in recipe.

Retuning restores Full's gain without reversing the Block–Full ordering. At the $1\times$ and $4\times$ budgets, the complete seed-9 learning-rate grids select 2.4×10^{-3} as the best tested rate for the baseline, Full, and Block separately, by endpoint validation loss; the next higher rate worsens all three (Table 4). At $4\times$, Full's gain is then 0.0311 nats and Block's is 0.0507. Repeating the selected rate on seed 1337 gives gains of 0.0159 and 0.0479 (Figure 3b). Both variants' gains relative to their retuned baselines are smaller at $4\times$ than at $1\times$, but Full's near-disappearance under the default recipe is absent. Block remains ahead at both endpoint budgets on both seeds. Its margin is nevertheless recipe-dependent:

Table 4: Learning rate × training budget at 124M (seed 9). Each cell is a complete cosine schedule; Δ compares the same-rate baseline. †: both depth-attention arms underperform the baseline. Additional seeds appear in Supplementary Table S9.

Budget	Peak LR	Baseline	Δ Full	Δ Block	$L_{\text{Full}} - L_{\text{Block}}$
1×	6×10^{-4} (default)	3.25292	+0.0314	+0.0511	+0.020
1×	1.2×10^{-3}	3.16798	+0.0269	+0.0516	+0.025
1×	$\mathbf{2.4 \times 10^{-3}}$ (best)	**3.13761**	**+0.0501**	**+0.0768**	**+0.027**
1×	4.8×10^{-3} †	3.15325	−0.2236	−0.1560	+0.068
2×	6×10^{-4}	3.09146	+0.0050	+0.0326	+0.028
2×	1.2×10^{-3}	3.05098	+0.0307	+0.0464	+0.016
4×	6×10^{-4} (default)	2.98975	−0.0002	+0.0323	+0.032
4×	1.2×10^{-3}	2.96181	+0.0348	+0.0376	+0.003
4×	$\mathbf{2.4 \times 10^{-3}}$ (best)	**2.94102**	**+0.0311**	**+0.0507**	**+0.020**
4×	4.8×10^{-3} †	2.95988	−0.1444	−0.0206	+0.124

at $4\times$ on seed 9, it is 0.003 nats at the intermediate rate, 0.032 at the default rate, and 0.020 at the best tested rate.

The 350M learning-rate ladder corroborates this ordering at the larger scale (Supplementary Table S10). Block leads Full at every tested rate, with margins of 0.017–0.035 nats before the highest-rate degradation. Again, 2.4×10^{-3} minimizes validation loss for all three architectures among the tested rates; there, Block improves on the baseline by 0.0621 nats and Full by 0.0268. This is a single-seed comparison at one training budget, so it extends the learning-rate check across scale without establishing how the budget trend transfers to 350M.

The distinction here is between an architectural ranking that survives retuning and an explanation of that ranking that does not. Block remains ahead of Full after retuning, whereas the apparent disappearance of Full's gain does not. Fixed-recipe budget scaling can therefore confound architectural persistence with recipe-dependent optimization effects. Appendix Figure S6 shows these patterns developing over training rather than only at the final evaluation. Claims that an architectural gain persists or vanishes with longer training are therefore substantially more informative when the architecture and its reference are retuned at the new budget. This conclusion is limited to the fourfold range at 124M, reaching approximately 80 tokens per parameter; the $2\times$ budget has only a partial learning-rate grid, and the best tested rate was not evaluated there.

6 Conclusion and Future Work

We studied whether finer-grained residual access is actually preferable in Attention Residuals. Across 124M and 350M language models, Block AttnRes consistently outperforms Full under the standard recipe, remains ahead after learning-rate retuning and longer training, and reduces measured GPU time per token by 14–26%. Controlled grouping experiments further show that candidate count alone is insufficient: local non-aligned and shifted groupings fail to recover Block's gains, while summaries spanning two complete Transformer blocks nearly match one-block summaries. Together, these results suggest that how residual history is partitioned is itself an inductive bias, with complete Transformer blocks providing a useful aggregation unit for depth-wise attention. They also show that conclusions about architectural persistence across training budgets can change after retuning, even when the architecture ranking does not. Several questions remain open. Our experiments stop at 350M parameters and roughly 80 training tokens per parameter, and the mechanism behind the alignment advantage is unresolved. Future work should test larger models and longer training, construct even more tightly matched grouping controls, and use routing interventions to determine why block-aligned summaries are easier for later layers to use.

AI use statement

In this work, we used a generative AI for executing and debugging experiment code, and for editing the prose of this manuscript. Research questions, experimental design, control construction and

all claims were decided by the authors. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

Ethics statement

This work studies the training dynamics of small language models on a public web-text corpus. It involves no human subjects, no private data and no release of a new dataset. We do not foresee ethical concerns specific to the architectural findings reported here beyond those shared by language modelling research in general.

Reproducibility statement

All models, data pipelines, groupings and the fused kernel are implemented in a single codebase that will be released upon acceptance of the paper, together with the exact partitions used for every grouping control, the configuration and details of all 113 training runs.

References

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in artificial intelligence*, pp. 1352–1361. PMLR, 2021.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2(3):5, 2018.

Irwan Bello, William Fedus, Xianzhi Du, Ekin Dogus Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting resnets: Improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 34:22614–22627, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International conference on machine learning*, pp. 2286–2296. PMLR, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Mike Heddes, Adel Javanmard, Kyriakos Axiotis, Gang Fu, MohammadHossein Bateni, and Vahab Mirrokni. Deepcrossattention: Supercharging transformer residual connections. *arXiv preprint arXiv:2502.06785*, 2025.

Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1412–1421, 2015.

Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*, 2018.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *Advances in neural information processing systems*, 37:136479–136508, 2024.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Falong Shen, Rui Gan, and Gang Zeng. Weighted residuals for very deep networks. In *2016 3rd international conference on systems and informatics (Icsai)*, pp. 936–941. IEEE, 2016.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *Advances in neural information processing systems*, 28, 2015.

Jonathan Su. Low-rank attention residuals. *arXiv preprint arXiv:2607.09694*, 2026.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 32–42. IEEE, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 1810–1822, 2019.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, et al. mhc: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pp. 10524–10533. PMLR, 2020.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pp. 2048–2057. PMLR, 2015.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Yilang Zhang, Bingcong Li, Niao He, and Georgios B Giannakis. Ancre: Adaptive neural connection reassignment for efficient depth scaling. *arXiv preprint arXiv:2602.09009*, 2026.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations*, volume 2025, pp. 97183–97219, 2025.

A Supplementary Quality and Cost Results

A.1 Default recipe including the static-skeleton arm

Table S1: Default-recipe comparison including the static-skeleton arm (Sparse+Sink, $k = 8$ wired sources plus a sink; Appendix A.1). Validation loss in nats; Δ is the same-seed baseline minus the arm, and the last column is Full minus Block (positive favours Block).

Scale	Seed	Baseline	Full	Δ	Sparse+Sink	Δ	Block	Δ	Full − Block
124M	1337	3.27251	3.24267	+0.0298	3.24830	+0.0242	3.21932	+0.0532	+0.0234
124M	8	3.27788	3.27222	+0.0057	3.26676	+0.0111	3.23305	+0.0448	+0.0392
124M	9	3.27742	3.24952	+0.0279	3.26389	+0.0135	3.22470	+0.0527	+0.0248
124M	mean			+0.0211		+0.0163		+0.0502	+0.0291
350M	1337	2.88258	2.87055	+0.0120	2.88019	+0.0024	2.85036	+0.0322	+0.0202
350M	8	2.87957	2.85256	+0.0270	2.87389	+0.0057	2.84549	+0.0341	+0.0071
350M	mean			+0.0195		+0.0040		+0.0332	+0.0136

Supplementary Table S1 repeats Table 2 with a fourth arm. *Sparse+Sink* is a static-skeleton reader: each sublayer input is one softmax over $k = 8$ statically wired candidates plus a *sink* candidate equal to the mean of all remaining candidates, maintained as a running sum, so a reader touches $k+1$ sources instead of all l available ones (readers with at most k candidates have no sink). The sink logit carries a fixed bias of $\log(l - k)$, so at zero initialization the weights are exactly $1/l$ on each wired candidate and $(l - k)/l$ on the sink, the uniform average over all l candidates, and function preservation holds as for the other arms (Appendix D.5). The wiring of each reader is the k candidates with the largest mean routing weight in a Full model trained at the same scale under the default recipe (seed 1337), so the arm tests whether a skeleton distilled from Full's learned routing can replace input-dependent access to every candidate. At these scales it cannot: it improves on the baseline at every seed (mean $+0.0163$ nats at 124M and $+0.0040$ at 350M) but trails Block in every cell and Full in four of five, the exception being 124M seed 8, where Full itself gains only $+0.0057$. A 350M variant that transfers the 124M wiring by depth-normalized index mapping instead of extracting it at 350M ended at 2.89058 on seed 1337, below the baseline.

A.2 Main-matrix trajectories

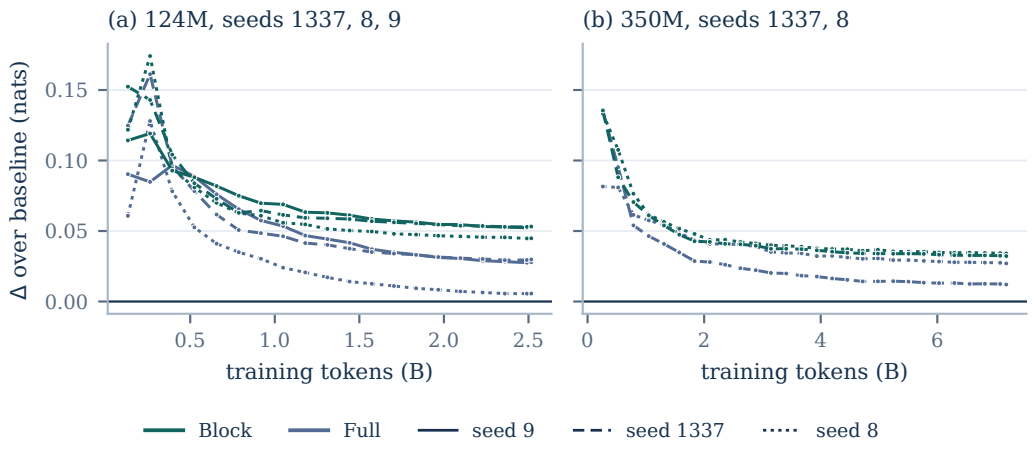


Figure S1: **Paired validation gain during training, main matrix.** $\Delta(t) = L_{\text{baseline}}(t) - L_{\text{arm}}(t)$ at shared evaluation steps for Full and Block at the standard budget and default recipe: (a) 124M, seeds 1337, 8 and 9; (b) 350M, seeds 1337 and 8. Line style encodes the seed.

Figure S1 gives the paired gains of the main matrix over training. By about 0.8B tokens, Block's gain exceeds Full's at every subsequent measured validation point in all five runs across the two scales. The endpoint values are those of Table 2.

A.3 COST AUDIT

Table S2: GPU time per token relative to the same-seed baseline, using seed 9 at 124M and seed 1337 at 350M. Ratios use steady-state median step times with the window defined in Supplementary Table S3; all depth-attention arms use fused Triton kernels. At 350M the baseline uses one B200 and depth-attention arms use two. Ratios account for GPU count and exclude evaluation, checkpointing and queueing.

Arm	124M, H100	350M, B200
Full	1.48×	2.14×
Sparse+Sink k8	1.31×	1.40×
Block	**1.28**×	**1.58**×

Table S3: Step-time audit for the main-matrix runs, using Figure 1's complete ten-update logging intervals after step 300 (449 intervals at 124M; 1,339 at 350M). Step time is the median interval-average time per update; Q1–Q3 are inclusive empirical quartiles. GPU time per token is GPUs × step time ÷ 524,288 tokens, relative to the same-seed baseline; ratios to Full compare equal GPU counts. Training hours are steps × median step time, excluding evaluation, checkpointing and queueing.

Scale	Arm	Seed	GPUs	Step (s)	Q1–Q3 (s)	GPU-time vs. baseline	Step vs. Full	Train h
124M	Baseline	1337	1	3.62	3.61–3.62	1.00	—	4.8
124M	Full	1337	1	5.34	5.33–5.36	1.47	1.000	7.1
124M	Block	1337	1	4.58	4.58–4.60	1.27	0.859	6.1
124M	Baseline	8	1	3.61	3.61–3.61	1.00	—	4.8
124M	Full	8	1	5.32	5.30–5.35	1.47	1.000	7.1
124M	Block	8	1	4.63	4.62–4.65	1.28	0.870	6.2
124M	Baseline	9	1	3.62	3.61–3.62	1.00	—	4.8
124M	Full	9	1	5.34	5.34–5.34	1.48	1.000	7.1
124M	Block	9	1	4.64	4.64–4.64	1.28	0.868	6.2
350M	Baseline	1337	1	4.04	4.04–4.04	1.00	—	15.4
350M	Full	1337	2	4.31	4.31–4.31	2.14	1.000	16.4
350M	Block	1337	2	3.19	3.19–3.19	1.58	0.741	12.2
350M	Baseline	8	1	4.05	4.05–4.05	1.00	—	15.4
350M	Full	8	2	4.31	4.31–4.31	2.13	1.000	16.4
350M	Block	8	2	3.19	3.19–3.19	1.58	0.741	12.1

Supplementary Table S3 gives the per-run steady-state medians behind Figure 1. Supplementary Table S2 also includes Sparse+Sink, using the same timing window on seed 9 at 124M and seed 1337 at 350M. Its GPU-time ratios compare each arm with the same-seed baseline.

A.4 EPSILON CONTROL

At reader k, zero-query readout scales the residual stream by $1/G_k$, the inverse number of active candidates; Full and Block have different G_k at almost every reader ($2L{+}1$ versus $L{+}1$ at the final reader), so the effective epsilon $G_k^2\epsilon$ of the consuming norm differs between the two arms (Section 3). We reran Full and Block at 124M (seed 9, default recipe, data sample A) with $\epsilon = 0$ in every RMSNorm of the model, that is the sublayer pre-norms, the output norm and the readers' key normalization, which a single configuration field controls. Final validation losses were 3.24861 (Full) and 3.22163 (Block), against 3.24952 and 3.22470 at the default epsilon: a Block–Full gap of 0.0270 nats against 0.0248.

Table S4: Granularity sweep at 124M (seed 9, default recipe, baseline 3.27742). *Blocks per summary* is the number of consecutive transformer blocks summed into one candidate; Full keeps every sublayer output separate.

Grouping	Candidates	Val. loss	Δ
Full (every sublayer separate)	25	3.24952	+0.0279
1 block per summary (Block)	13	**3.22470**	**+0.0527**
2 blocks per summary	7	3.22595	+0.0515
3 blocks per summary	5	3.23501	+0.0424
4 blocks per summary	4	3.24400	+0.0334

Table S5: Paired validation-loss differences (nats). "A over B" denotes $L_B - L_A$, so positive values favour A. Columns cover the 124M default and tuned recipes (seeds 9, 1337) and the 350M default recipe (seed 1337). *Local* is the local non-aligned random control, *shifted* the half-block shifted control, and *global* the global random pairs. Local matches Block's final candidate count; shifted has one extra candidate. The controls also differ in reader exposure (Table 1), so these contrasts do not isolate alignment, contiguity or depth locality. Local over Full jointly changes candidate count and grouping.

	124M default		124M tuned		350M	
Contrast	s9	s1337	s9	s1337	s1337	Positive / measured
Block over local	+0.019	+0.020	+0.018	+0.022	**+0.015**	5/5
Block over shifted	+0.026	+0.017	+0.015	+0.006	—	4/4
Local over Full	+0.006	+0.003	+0.009	−0.009	+0.006	4/5
Shifted over local	−0.007	+0.003	+0.003	+0.015	—	3/4
Local over global	+0.033	+0.029	+0.024	+0.024	+0.006	5/5
Block over Full	+0.025	+0.023	+0.027	+0.013	**+0.020**	5/5

B Supplementary Grouping Analyses

B.1 Granularity and paired grouping differences

Supplementary Table S4 is the granularity sweep behind Figure 2c, and Supplementary Table S5 lists the paired grouping differences of Section 5.2 cell by cell.

B.2 Reader exposure

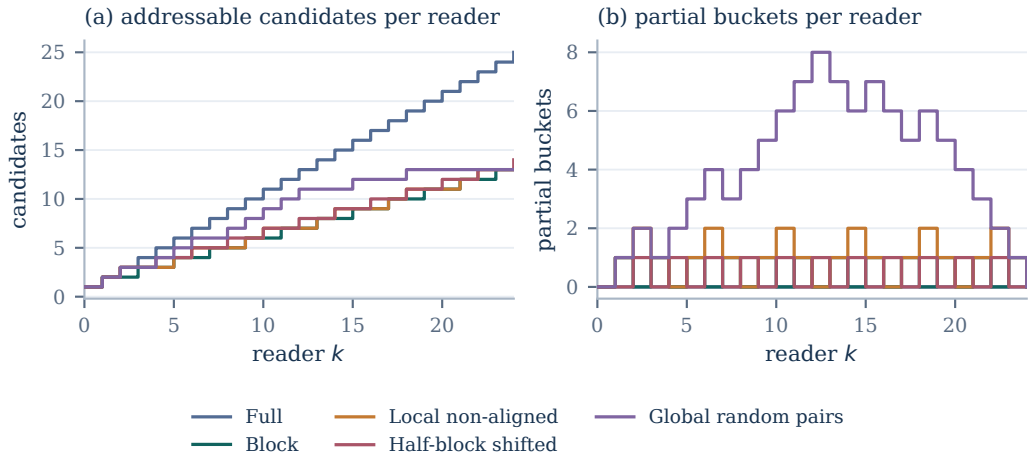


Figure S2: **Reader exposure by grouping**, 12-layer model: (a) number of addressable candidates and (b) number of partial buckets seen by reader k (the input to sublayer $k{+}1$; $k = 24$ is the final readout). Means over readers are given in Table 1. These are properties of the partitions, not measurements.

Table S6: Reader-exposure audit for the 24-layer model (350M), the analogue of Table 1: mean number of addressable candidates and partial buckets over the $2L+1 = 49$ readers, aligned pairs in the final partition, and whether the grouping was trained at 350M. The shifted partition is listed for completeness.

Grouping	Final buckets	Cand. / reader	Partial / reader	Aligned pairs	Run at 350M
Block (aligned)	25	13.24	0.49	24	yes
Local non-aligned random	25	13.49	0.98	0	yes
Half-block shifted	26	13.73	0.47	0	no
Global random pairs	25	17.39	8.78	1	yes
Full	49	25.00	0.00	—	yes

Figure S2 expands Table 1 reader by reader. Supplementary Table S6 repeats the audit for the 24-layer model used at 350M: the local control again sits close to Block (13.49 versus 13.24 candidates, 0.98 versus 0.49 partial buckets per reader) while global random pairing does not (17.39 and 8.78). The local and shifted controls track Block within one candidate at every reader; the global random pairing exposes intermediate readers to up to four more candidates at 12 layers and seven at 24 layers, most of them partial.

B.3 Grouping dynamics

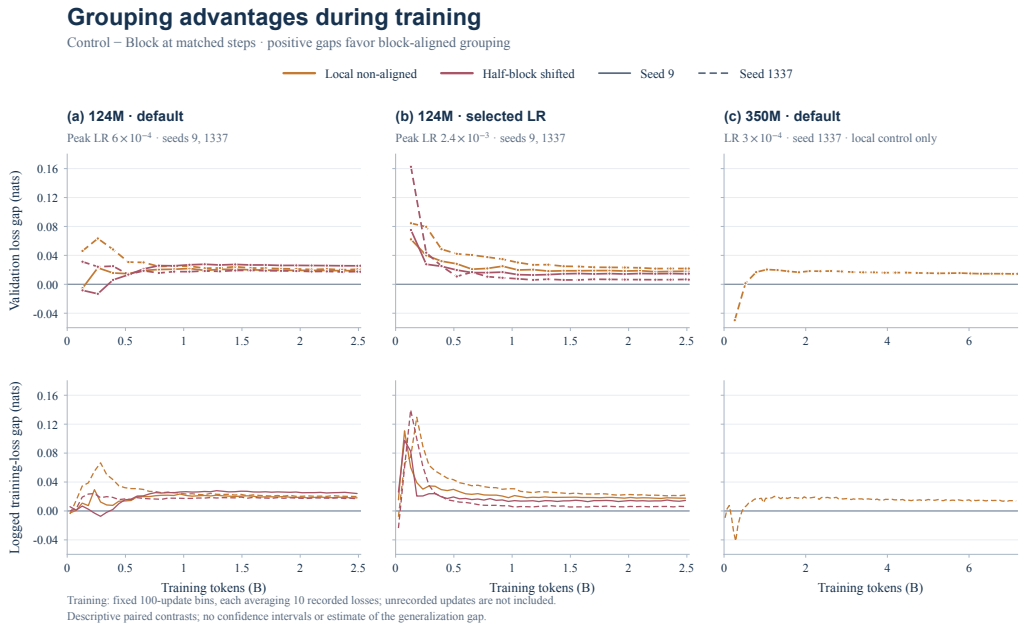


Figure S3: **Paired grouping contrasts during training**, control minus Block at matched steps (positive favours Block), for the local non-aligned (orange) and half-block shifted (rose) controls under (a) the 124M default recipe, seeds 9 and 1337; (b) the 124M selected rate, seeds 9 and 1337; (c) the 350M default recipe, seed 1337, local control only. Top row: validation-loss gap at shared evaluation steps. Bottom row: gap of logged training losses averaged in fixed 100-update bins. Descriptive paired contrasts without confidence intervals.

Figure S3 tracks the grouping contrasts of Section 5.2 through training. From about 1B tokens onward, Block's advantage over the available matched controls remains positive in every condition, although some margins continue to narrow. The endpoint values are those of Supplementary Table S5.

Table S7: Candidate statistics for 124M models (seed 9, default recipe). Each entry averages the per-token statistic over evaluation tokens and all pairs of the indicated type: 12 *aligned* pairs $(\boldsymbol{a}_i, \boldsymbol{m}_i)$, 11 *shifted* pairs $(\boldsymbol{m}_i, \boldsymbol{a}_{i+1})$, or the 12 *random* pairs of the global random partition. The same pairings are evaluated on every model. The audited shifted model folds the embedding into its first bucket (13 candidates); its 11 measured sublayer pairs also occur in the main half-block shifted control (14 candidates).

Model audited	Pairing	Mean $\|u+v\|$	Mean $\cos(u, v)$
Full	aligned	19.10	0.145
	shifted	18.73	0.127
	random	18.49	0.085
Block	aligned (merged pairs)	16.33	0.101
	shifted	16.60	0.055
	random	15.76	0.055
Shifted (embedding folded)	aligned	15.08	0.088
	shifted (merged pairs)	14.46	0.109
	random	13.96	0.083
Global random	aligned	14.90	0.071
	shifted	13.26	0.039
	random (merged pairs)	15.53	**0.177**

B.4 Candidate-magnitude audit

Equal bucket sizes need not give equal candidate norms: $\|u + v\|^2 = \|u\|^2 + \|v\|^2 + 2\langle u, v\rangle$. Supplementary Table S7 reports held-out means of pre-aggregation pair-sum norms and cosines. Each grouping-trained model has the highest mean cosine for its own pairs, yet the global-random model has the worst loss among these models. These endpoint associations do not isolate magnitude effects: RMS-normalized keys retain epsilon-dependent scale sensitivity, and values remain unnormalized. The audited shifted model folds the embedding into its first bucket, unlike the main shifted control. Statistics are means over the token positions of held-out validation batches at the final checkpoint; the pairs are the aligned $(\boldsymbol{a}_i, \boldsymbol{m}_i)$, shifted $(\boldsymbol{m}_i, \boldsymbol{a}_{i+1})$ and global-random pairs of Supplementary Table S14, evaluated on every model whether or not that model merges them.

B.5 Learned routing

Protocol. For each of the four models (124M, seed 9, default recipe, final checkpoint) we run eight held-out batches of eight 2,048-token sequences from the model's training sample, 131,072 token positions in total, and record for every reader its softmax weights over the candidates it can address, averaged over all token positions and batches. Panels (a–d) of Figure S4 show these averaged weights in slot order and panel (e) the averaged weight on the embedding candidate. Panel (f) is the normalized entropy $H_k / \log n_k$ of the averaged weight vector of reader k, with n_k the number of candidates that reader can address, so that groupings with different candidate counts are comparable; it is the entropy of the token-averaged distribution, not the average per-token entropy, and reader 0, which has a single candidate, is excluded from the means quoted below.

Figure S4 shows the learned routing of the four 124M seed-9 models compared in Section 5.2. The gross structure is shared. Every reader begins by reading the embedding, whose weight falls to 0.20–0.23 by reader 12 and to 0.05–0.10 at the final readout for all four groupings; later readers concentrate on the most recently produced candidates; and the normalized entropy of the weights, averaged over readers 1–24, lies between 0.73 (Full) and 0.79 (Block), so Block's routing is, if anything, closer to uniform over its candidates than Full's is over its own. The groupings differ in what the final readout favours: Block and the local control put their largest final weight on the most recent pair (0.33 and 0.34), whereas Full and the shifted control put theirs on the first attention output $\boldsymbol{a}_1$ (0.24 and 0.40). The figure is descriptive: it characterizes what the trained readers attend to under each grouping and does not test why the block-aligned grouping is preferred.

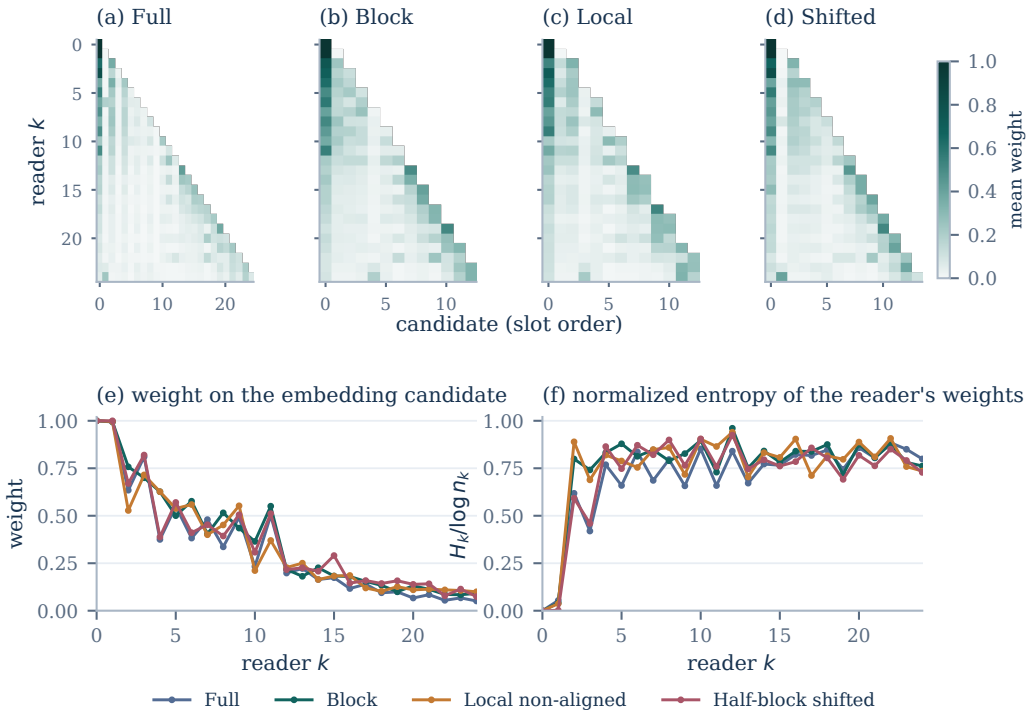


Figure S4: **Learned routing at 124M** (seed 9, default recipe, final checkpoints, averaged over eight validation batches). (a–d) Mean attention weight of every reader k over its candidates in slot (first-produced) order for Full, Block, the local non-aligned control and the shifted control; cells right of the staircase are candidates not yet produced for that reader. (e) Weight each reader places on the embedding candidate. (f) Normalized entropy of each reader's weights, $H_k / \log n_k$ with n_k the number of candidates the reader can address (1 is uniform). Descriptive only.

Table S8: Grouping sweep at 124M (seed 9, default recipe, baseline 3.27742), ordered from highest to lowest validation loss. *Candidates* is the number of buckets at the final reader. Each row is a single-seed measurement.

Grouping	Candidates	Δ	Note
Global random pairs	13	+0.0009	matches Block's final size profile
Global random quads	7	+0.0024	
Same-type bands (attn / MLP)	7	+0.0139	groups four same-type outputs
α-profile greedy merge	7	+0.0207	uses Full's routing profiles
Telescope, reversed (fine distant)	7	+0.0233	
Half-block shifted, embedding folded	13	+0.0270	
Half-block shifted, embedding separate	14	+0.0270	
Full	25	+0.0279	reference
Telescope (fine recent)	7	+0.0308	
Uniform, 4 blocks per summary	4	+0.0334	
Local non-aligned random	13	+0.0336	close to Block in reader exposure
Uniform, 3 blocks per summary	5	+0.0424	
Uniform, 2 blocks per summary	7	+0.0515	loss 0.00125 nats above Block
Uniform, 1 block per summary (Block)	13	**+0.0527**	lowest observed loss

B.6 FULL GROUPING MAP AT 124M

Supplementary Table S8 lists every grouping trained at 124M (seed 9, default recipe) in order of validation loss.

C Supplementary Budget and Learning-Rate Results

C.1 Budget endpoints across seeds

Table S9: Selected budget endpoints at 124M, covering every measured seed. Each row is a complete cosine schedule on the second data sample; Δ uses the same-budget, same-rate, same-seed baseline. The best tested rate was selected on seed 9. The 2× budget covers only seed 9 at the two shown rates.

Budget	Peak LR	Seed	Baseline	Δ Full	Δ Block	$L_{\text{Full}} - L_{\text{Block}}$
1×	6×10^{-4} (default)	9	3.25292	+0.0314	+0.0511	+0.0197
1×	6×10^{-4} (default)	1337	3.24124	+0.0136	+0.0427	+0.0291
1×	6×10^{-4} (default)	8	3.24975	+0.0125	+0.0423	+0.0297
1×	2.4×10^{-3} (best tested)	9	3.13761	+0.0501	+0.0768	+0.0267
1×	2.4×10^{-3} (best tested)	1337	3.12959	+0.0383	+0.0515	+0.0132
2×	6×10^{-4} (default)	9	3.09146	+0.0050	+0.0326	+0.0276
2×	1.2×10^{-3}	9	3.05098	+0.0307	+0.0464	+0.0158
4×	6×10^{-4} (default)	9	2.98975	-0.0002	+0.0323	+0.0325
4×	6×10^{-4} (default)	1337	2.98907	+0.0053	+0.0331	+0.0278
4×	6×10^{-4} (default)	8	2.99312	-0.0042	+0.0256	+0.0298
4×	2.4×10^{-3} (best tested)	9	2.94102	+0.0311	+0.0507	+0.0196
4×	2.4×10^{-3} (best tested)	1337	2.94151	+0.0159	+0.0479	+0.0320

Supplementary Table S9 lists selected 124M budget endpoints covering all three measured seeds; Table 4 gives the full seed-9 grids at $1\times$ and $4\times$. Block improves on the baseline and on Full in all twelve cells. Full's gain at the default rate and the $4\times$ budget is within 0.006 nats of zero on all three seeds (-0.0002, $+0.0053$, -0.0042), whereas at the best tested rate it is $+0.0311$ and $+0.0159$; the shrinking of Full's gain with budget discussed in Section 5.3 is therefore a default-rate observation on every seed that was run, not a property of one seed.

C.2 Learning-rate response

Table S10: Learning-rate ladder at 350M (seed 1337, B200). Default peak rate is 3×10^{-4}. At 4.8×10^{-3} the baseline becomes unstable (†), so Δ is not meaningful there and is shown in parentheses.

Peak LR	Baseline	Full	Δ Full	Block	Δ Block	$L_{\text{Full}} - L_{\text{Block}}$
3×10^{-4} (default)	2.88258	2.87055	+0.0120	2.85036	+0.0322	+0.0202
6×10^{-4}	2.83101	2.82275	+0.0083	2.78878	+0.0422	+0.0340
1.2×10^{-3}	2.80159	2.75726	+0.0443	2.74059	+0.0610	+0.0167
$\mathbf{2.4 \times 10^{-3}}$ (best)	**2.77833**	**2.75153**	**+0.0268**	**2.71624**	**+0.0621**	**+0.0353**
4.8×10^{-3}	7.39775†	3.73216	(+3.666)	3.61339	(+3.784)	+0.1188

Supplementary Table S10 is the 350M learning-rate ladder discussed in Section 5.3, and Figure S5 plots the endpoint losses behind it and Table 4. At both scales and both 124M budgets the rate above 2.4×10^{-3} worsens all three arms, so the best tested rate is interior to the grid for every arm, and Block's curve lies below Full's at every rate at which both converge.

C.3 Budget trajectories

Figure S6 plots the paired validation gain during the $4\times$ 124M runs at the default and retuned learning rates, complementing the endpoint comparison in Section 5.3.

C.4 Behaviour past the best tested rate

Figure S7 shows the trajectories behind Supplementary Table S11. At 124M one doubling past the best tested rate costs the baseline 0.016 nats but Full 0.289 and Block 0.248; at 350M the same

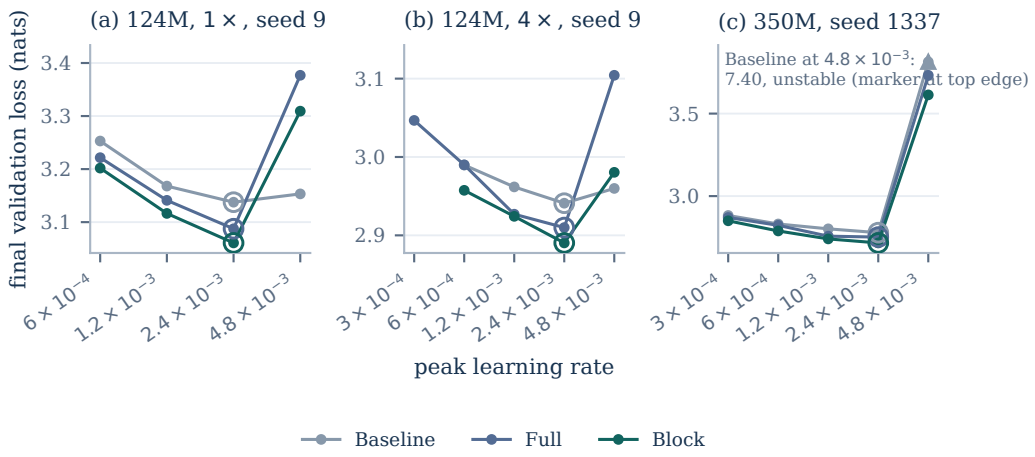


Figure S5: **Learning-rate response.** Final validation loss against peak learning rate for the baseline, Full and Block at (a) 124M, standard budget, seed 9; (b) 124M, 4× budget, seed 9; (c) 350M, standard budget, seed 1337. Each point is a complete cosine schedule. Rings mark each arm's best tested rate. In (b) Full has an additional run at 3×10^{-4}. The 350M baseline at 4.8×10^{-3} was unstable; its final loss is printed and the marker sits at the top edge.

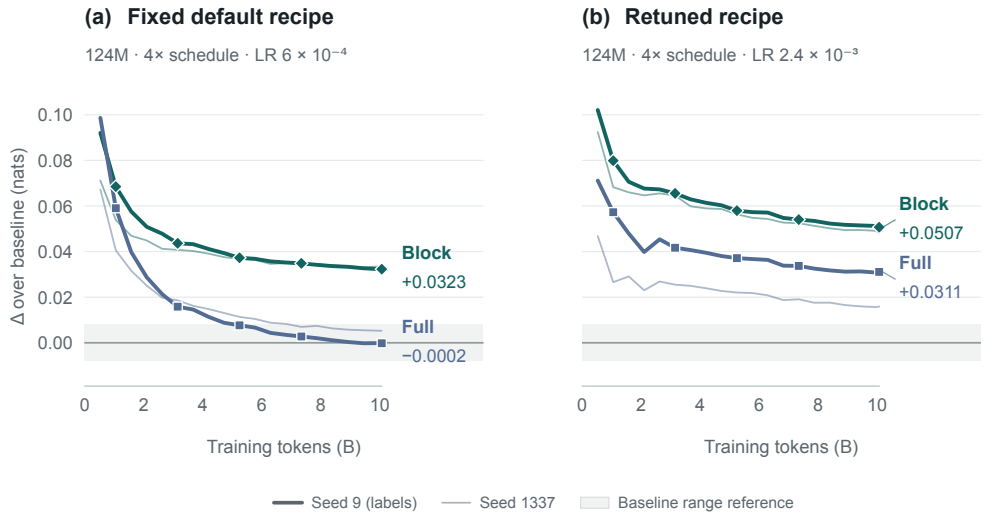


Figure S6: **Paired validation-gain trajectories:** $\Delta(t) = L_{\text{baseline}}(t) - L_{\text{arm}}(t)$ at shared evaluation steps during the 4× 124M schedule, at (a) the default and (b) the retuned peak learning rate; thick lines with endpoint labels are seed 9, thin lines seed 1337. The shaded band is the same baseline reference range as in Figure 3. These are trajectories of single completed runs and complement the endpoint comparison of Figure 3; they are not checkpoints of one longer run.

Table S11: Loss added by doubling the best tested peak learning rate (2.4×10^{-3} to 4.8×10^{-3}), in nats: seed 9 at 124M and seed 1337 at 350M's standard budget. At 124M the depth-attention arms degrade more; at 350M the baseline degrades much more.

	Baseline	Full	Block
124M, 1× budget	+0.016	+0.289	+0.248
124M, 4× budget	+0.019	+0.194	+0.090
350M	**+4.619 (unstable)**	+0.981	+0.897

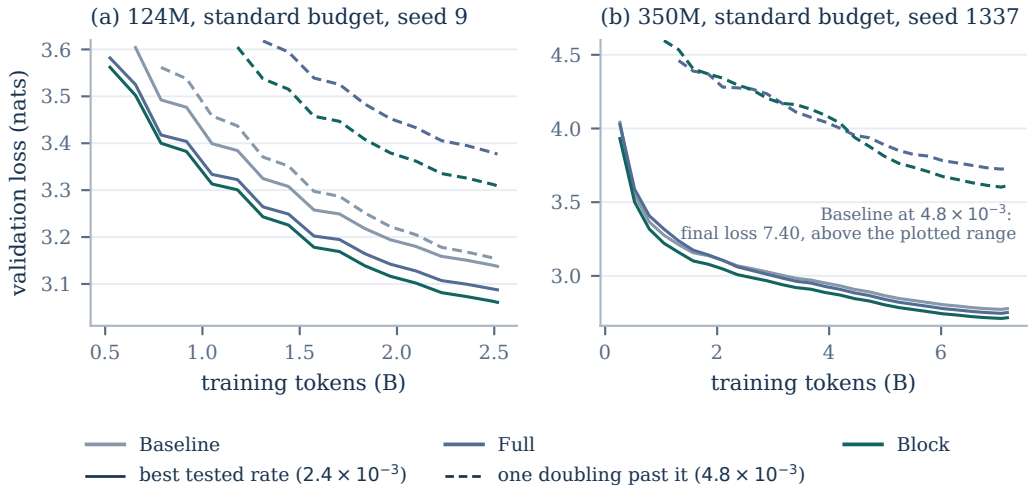


Figure S7: **One doubling past the best tested rate.** Validation loss during training at each arm's best tested rate (2.4×10^{-3}, solid) and one doubling past it (4.8×10^{-3}, dashed): (a) 124M, standard budget, seed 9; (b) 350M, standard budget, seed 1337. Only the part of each curve inside the plotted range is drawn, so early high-loss segments enter from the top; the 350M baseline at 4.8×10^{-3} never enters the range and its final value is printed.

doubling causes severe instability in the baseline, which ends at 7.40 nats, while Full and Block end at 3.73 and 3.61 nats.

D REPRODUCIBILITY DETAILS

D.1 CONFIGURATIONS

Table S12: Configurations. Budget and learning-rate runs change the peak learning rate, steps (with evaluation and checkpoint intervals scaled) and data sample. Attention and MLP output projections use the base initialization std divided by $\sqrt{2L}$; depth queries start at zero and norm gains at one. Weight decay applies to matrix parameters; vectors have no decay. AdamW uses $\epsilon = 10^{-8}$.

	124M	350M
Blocks / width / heads	12 / 768 / 12	24 / 1024 / 16
FFN width / activation	2048 / SwiGLU	2816 / SwiGLU
Attention	12 heads, causal, no KV sharing	16 heads, same
Tokenizer	GPT-2 BPE, 50,257 padded to 50,304	same
Vocabulary / tied embeddings	50304 / yes	50304 / yes
Sequence length / RoPE θ	2048 / 10000	2048 / 10000
Base init std	0.02	0.02
Global batch (sequences / tokens)	256 / 524,288	256 / 524,288
Standard budget (steps / tokens)	4,800 / 2.5B	13,700 / 7.2B
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$)	same
Default peak LR / min ratio	6e-4 / 0.1	3e-4 / 0.1
Warm-up steps	200	300
Weight decay / grad clip (ℓ_2 norm)	0.1 / 1.0	0.1 / 1.0
RMSNorm ϵ (all norms)	10^{-6}	10^{-6}
Precision	bf16 autocast, fp32 master weights	same
Parameters (baseline / arms)	123,587,328 / 123,625,728	359,842,816 / 359,943,168
Eval interval / eval tokens	250 / 10,485,760	500 / 10,485,760
Checkpoint interval	1200	2000
Depth-attention kernel / key norm	softmax / RMSNorm	same
Depth-attention implementation	fused Triton kernel	fused Triton kernel
Hardware (baseline / depth arms)	1×H100 / 1×H100	1×B200 / 2×B200

Supplementary Table S12 lists the model and optimizer settings shared by all runs at each scale.

D.2 DATA SAMPLES

Table S13: FineWeb-Edu samples by run family. Sample A is the smaller sample used for the 124M standard-recipe campaign; sample B is a larger sample used for the 124M budget and learning-rate campaign and for all 350M runs. Every paired difference in the paper is taken within one sample.

Runs	Sample	Seeds
124M main matrix	A (fineweb_edu)	1337, 8, 9
124M granularity sweep; epsilon control	A	9
124M default-recipe grouping controls	A	9; selected controls also 1337
124M budget × learning-rate grid	B (fineweb_edu_100bt)	9; 1337 and 8 at selected endpoints
124M tuned-recipe grouping controls	B	9; selected controls also 1337
350M main matrix	B	1337, 8
350M learning-rate ladder; grouping controls	B	1337

Supplementary Table S13 records which FineWeb-Edu sample each family of runs was trained on; every paired difference in the paper is taken within one sample.

Table S14: Realized partitions. a_i and m_i are the attention and MLP outputs of block i; *emb* is the token embedding. An aligned pair counts only a two-member bucket $\{a_i, m_i\}$, excluding complete blocks inside larger buckets. Random partitions were drawn once with an independent seed and reused across training seeds and recipes.

Local non-aligned (12 layers) (13 buckets, 0 aligned pairs)
{emb} {a1,m2} {m1,a2} {a3,m4} {m3,a4} {a5,a6} {m5,m6} {a7,m8} {m7,a8} {a9,a10} {m9,m10} {a11,m12} {m11,a12}

Half-block shifted (12 layers) (14 buckets, 0 aligned pairs)
{emb} {a1} {m1,a2} {m2,a3} {m3,a4} {m4,a5} {m5,a6} {m6,a7} {m7,a8} {m8,a9} {m9,a10} {m10,a11} {m11,a12} {m12}

Half-block shifted, embedding folded (12 layers) (13 buckets, 0 aligned pairs)
{emb,a1} {m1,a2} {m2,a3} {m3,a4} {m4,a5} {m5,a6} {m6,a7} {m7,a8} {m8,a9} {m9,a10} {m10,a11} {m11,a12} {m12}

Global random pairs (12 layers) (13 buckets, 0 aligned pairs)
{emb} {a5,m7} {a1,a2} {m5,a12} {m4,a9} {m1,m12} {m2,m8} {a3,a4} {a8,m11} {a6,a7} {m3,a11} {m6,a10} {m9,m10}

Global random quads (12 layers) (7 buckets, 0 aligned pairs)
{emb} {a1,a2,a5,m7} {m4,m5,a9,a12} {m1,m2,m8,m12} {a3,a4,a8,m11} {m3,a6,a7,a11} {m6,m9,a10,m10}

Same-type bands (12 layers) (7 buckets, 0 aligned pairs)
{emb} {a1,a2,a3,a4} {a5,a6,a7,a8} {a9,a10,a11,a12} {m1,m2,m3,m4} {m5,m6,m7,m8} {m9,m10,m11,m12}

α-profile greedy merge (12 layers) (7 buckets, 0 aligned pairs)
{emb} {a1,a9,a11,a12} {m1,a4,m4,a5,a6,m6,a7,m7,a10} {a2,a3} {m2,m5,a8,m8,m9,m10,m11} {m3} {m12}

Local non-aligned (24 layers) (25 buckets, 0 aligned pairs)
{emb} {a1,m2} {m1,a2} {a3,m4} {m3,a4} {a5,a6} {m5,m6} {a7,m8} {m7,a8} {a9,a10} {m9,m10} {a11,m12} {m11,a12} {a13,a14} {m13,m14} {a15,m16} {m15,a16} {a17,m18} {m17,a18} {a19,a20} {m19,m20} {a21,m22} {m21,a22} {a23,a24} {m23,m24}

Global random pairs (24 layers) (25 buckets, 1 aligned pair)
{emb} {m1,m15} {m9,a22} {a11,m17} {a5,a10} {a3,a16} {a17,m21} {m8,m18} {m3,a24} {m4,a8} {m2,m16} {a1,a14} {m5,m24} {m6,a7} {a12,a21} {a2,a9} {a15,m22} {a4,m14} {m7,a20} {m10,m13} {a13,a23} {a6,m11} {a19,m19} {m12,m23} {a18,m20}

D.3 GROUPING PARTITIONS

Supplementary Table S14 gives the fixed partitions used by the grouping controls. Telescope instead rebuilds its candidates at each reader: the embedding stays separate, up to two most recent sublayer outputs remain singletons, and the remaining outputs are summed in contiguous groups of widths $2, 4, 8, \ldots$ toward the oldest output, truncating the last group to the available history. Telescope reversed retains up to two oldest outputs as singletons and grows the groups toward the newest output. Both use seven candidates at the final reader of the 12-layer model.

D.4 WALL-CLOCK

Table S15: Training-time estimates at the standard budget: GPUs, steps, the median across seeds of per-run median step times, and the range over seeds of steps $\times$ median step time. Per-run timing uses Supplementary Table S3's protocol. Evaluation, checkpointing and queue time are excluded.

Scale	Arm	GPUs	Steps	Step (s)	Train hours
124M	Baseline	1	4,800	3.62	4.8–4.8
124M	Full	1	4,800	5.34	7.1–7.1
124M	Block	1	4,800	4.63	6.1–6.2
350M	Baseline	1	13,700	4.04	15.4–15.4
350M	Full	2	13,700	4.31	16.4–16.4
350M	Block	2	13,700	3.19	12.1–12.2

Supplementary Table S15 converts the step-time audit into training hours per run at the standard budget.

D.5 IMPLEMENTATION AND VERIFICATION CHECKS

One kernel for every arm. Every depth-attention arm is the same softmax readout over key-normalized candidates, executed by one fused Triton kernel: Full wires every candidate; Block, the granularity variants and the grouping controls pass bucket sums as candidates; Sparse+Sink wires k candidates plus a sink maintained as a running sum. An eager PyTorch implementation of the same operation is kept for testing, and every step-time ratio in Supplementary Table S3 compares arms that use the fused path.

Function preservation. With every routing query zero-initialized, Full and the grouped readouts assign uniform weights to their active candidates, while Sparse+Sink uses its multiplicity bias to recover the uniform average over the underlying sources (Appendix A.1); in each case the resulting stream is the standard residual sum times a positive per-token scalar, and because every consumer of the stream is a scale-invariant RMSNorm, the logits coincide exactly when the norm's epsilon is zero. Unit tests in float64 with epsilon zero assert this to an absolute and relative tolerance of 10^{-8} on logits and loss for Full, Block at one, two and four blocks per summary, several explicit partitions, the telescoping grouping and Sparse+Sink at $k \in \{1, 2, 8\}$, and assert that a nonzero query changes the function. With the default epsilon of 10^{-6} the equality holds up to the epsilon term; Appendix A.4 measures its effect on trained models.

Kernel parity. In float32 the fused and eager paths agree on outputs and value gradients to 2×10^{-4} absolute and relative, and on the query gradient to a relative ℓ_2 error below 10^{-4}. With bf16 inputs, the training path, they agree to a relative ℓ_2 error below 2% on outputs and gradients, the two implementations rounding at different points. Parity is also checked end to end on a complete small model.

E EXPERIMENT INVENTORY

Table S16: Inventory of the 113 standardized training runs. The 350M matrix includes eight main runs (two seeds × four arms) and one transferred-wiring diagnostic. The learning-rate ladder adds four rates; its default-rate runs are counted in the matrix. Earlier hardware and implementation campaigns are excluded.

Campaign	Runs	Purpose
124M standardized matrix, 3 seeds × 4 arms	12	main comparison
124M granularity sweep	3	blocks per summary
124M epsilon = 0 control	2	normalization confound
124M grouping controls, default recipe	14	alignment, locality, resolution
124M grouping controls, tuned recipe	7	same, at the best tested rate
124M budget × learning-rate grid	52	recipe sensitivity
350M standardized matrix	9	8 main + 1 transfer diagnostic
350M learning-rate ladder	12	4 additional rates × 3 arms
350M grouping controls	2	grouping at 350M
Total	113	

Supplementary Table S16 counts the 113 standardized training runs by campaign. The 350M main-matrix category includes one additional Sparse+Sink transfer diagnostic; the learning-rate category counts only the four additional rates, since the default-rate runs are already counted in the main matrix. The 52-run 124M budget campaign comprises 16 matched three-arm conditions, three Sparse+Sink runs, and one Full-only diagnostic. Supplementary Table S17 reports the additional grouping and budget diagnostics.

Table S17: Additional endpoints completing the standardized-run inventory. All runs use 124M models on one H100; data samples A and B are defined in Supplementary Table S13. Budgets $1\times$, $2\times$ and $4\times$ are 4,800, 9,600 and 19,200 steps. Sparse+Sink uses $k = 8$ and the 124M wiring of Appendix A.1. The Full-only diagnostic, also plotted in Figure S5, has no baseline at its learning rate, so no paired gain is reported. Losses are endpoint validation cross-entropy in nats.

Arm	Sample	Seed	Budget	Peak LR	Final loss
Half-block shifted, embedding folded	A	1337	$1\times$	6×10^{-4}	3.24249
Telescope (fine recent)	A	1337	$1\times$	6×10^{-4}	3.23781
Half-block shifted, embedding folded	B	9	$1\times$	2.4×10^{-3}	3.10694
Sparse+Sink	B	9	$1\times$	6×10^{-4}	3.23829
Sparse+Sink	B	9	$2\times$	6×10^{-4}	3.10274
Sparse+Sink	B	9	$4\times$	6×10^{-4}	2.99401
Full (LR diagnostic)	B	9	$4\times$	3×10^{-4}	3.04667