Output-Anchored Sparse RMSNorm in Byte-Level Language Models

Anonymous ACL submission

Abstract

Where should a fixed sparse RMSNorm budget be spent inside small byte-level transformers? We find that an output-anchored every-other-block layout (SPARSE-FINAL) is the right choice in the small regime: it lowers validation loss relative to dense FULL-RMS from 8.6M through a four-seed 151.7M corroboration under standard AdamW, and in encoder-decoder models the same sparse budget works best on the decoder side rather than the encoder side. The advantage is bounded, however: four-seed 17×832 and 18×832 bridges bring SPARSE-FINAL to parity and then slightly below dense, and follow-up standard-recipe probes at 252.0M and 309.2M reproduce the same negative pattern, with dense ahead at 252.0M and sparse diverging at 309.2M. A matched four-seed μP recipe with gap gating restores stability and keeps a modest throughput edge over FULL-RMS+μP, but it does not recover the loss advantage, and longer-horizon and 512-token probes move the result further toward dense normalization. Our main takeaway is a placement-and-scale boundary, not a new replacement for dense normalization.

1 Introduction

Normalization is a default ingredient in Transformer language models, but where the surviving norms sit can change the residual geometry seen by the optimizer (Xiong et al., 2020; Shleifer and Ott, 2022; Kim et al., 2025). The familiar Pre-LN/Post-LN axis decides which side of the residual a norm occupies; here we study a different question: if only some norms survive, which blocks should keep them?

The answer is not uniform across scale. In a byte-level harness with matched tokenizer, vocabulary, packing, and loss, an output-anchored every-other-block layout improves over dense RMSNorm in small decoder-only models, and the same sparse budget prefers the decoder side over the encoder side in encoder-decoder models. But the small-regime gain narrows steadily as scale grows, reaches parity around the 17×832 bridge, and flips negative by 200M–309M under the standard recipe. A matched μP protocol and gap gating recover stability and a small throughput edge, yet they do not overturn dense normalization once we extend training or context.

Existing work motivates the setting but does not isolate it. RMSNorm can be redesigned, replaced, tapered, removed by fine-tuning, or substituted by careful initialization and residual scaling (Nguyen and Salazar, 2019; Bachlechner et al., 2021; Zhang et al., 2019; Huang et al., 2020; Heimersheim, 2024; Zhu et al., 2025; Kanavalau et al., 2026). None of those papers ask which subset of RMSNorm layers should survive once we have decided to keep some, which is the question we study here.

We therefore compare dense RMSNorm with output-anchored sparse periodic RMSNorm (SPARSE-FINAL), late sparse layouts, and higher-scale rescue variants based on μP and gap gating in decoder-only models, and we test decoder-side against encoder-side sparse placement in encoder-decoder models. The result is a clean boundary map for a sparse-norm budget: useful at small scale, stabilizable at larger scale, but not a drop-in replacement for dense normalization.

Our contributions are:

- A controlled study showing that output-anchored sparse periodic RMSNorm improves over dense FULL-RMS from 8.6M through a four-seed 151.7M decoder sweep, and that, with the same sparse budget, decoder-side placement beats encoder-side placement in the encoder-decoder family.
- A boundary localization at 17×832, 18×832, 252M, and 309M, identifying the parameter range in which the standard-AdamW sparse

advantage disappears and showing that a follow-up 252M/309M probe reproduces the same dense-favored pattern.

- A matched four-seed μP + gap-gating study that prevents the catastrophic high-scale failure and keeps a modest throughput edge, but that does not overturn the dense baseline once we extend training or context.

2 Related Work

Our work sits at the intersection of normalization placement, normalization reduction, and norm-free residual stabilization for language models. The Pre-LN versus Post-LN literature established that normalization placement materially affects optimization and gradient scale (Xiong et al., 2020). More recent output-side placement work reaches a similar conclusion in decoder language models by showing that branch-output normalization can control residual-scale growth at larger scale (Kim et al., 2025). DeepNet then showed that residual and normalization design remains a central constraint when pushing Transformers to much greater depth (Wang et al., 2024). NormFormer further showed that even adding normalization can improve pretraining, reinforcing the view that normalization placement is a meaningful optimization variable (Shleifer and Ott, 2022). We use Liu et al. (2020) later only as broader evidence that residual-branch amplification can destabilize Transformer optimization.

Several papers show that Transformers can tolerate less normalization than the default recipe suggests. RMSNorm replaces LayerNorm with a cheaper scaling-only alternative (Zhang and Sennrich, 2019). More recent work explores stronger departures, including dynamic replacements such as DyT, tapered removal schedules, and post-hoc removal by fine-tuning (Zhu et al., 2025; Kanavalau et al., 2026; Heimersheim, 2024). These papers motivate our setting, but they do not isolate the matched-budget placement question studied here.

There is also a broader literature on training deep residual networks with reduced or no normalization. ReZero, Fixup, T-Fixup, NFNets, and related initialization-driven methods show that optimization can be stabilized without conventional per-block normalization if residual scale is controlled elsewhere (Bachlechner et al., 2021; Zhang et al., 2019; Huang et al., 2020; Brock et al., 2021). Our setting is different in two ways: we keep the standard Transformer blocks and optimizer, and we ask where a small number of surviving RMSNorm layers should be placed, not how to eliminate normalization entirely.

3 Experimental Setup

Paper names. We use short paper-facing names in the main text: FULL-RMS for the dense RMSNorm baseline, SPARSE-FINAL for the output-anchored every-other-block sparse recipe, and GAP-GATED SPARSE for the same sparse layout with a learnable ReZero-style scalar in each unnormalized gap (Bachlechner et al., 2021). Additional sparse layouts appear only when needed and are mapped to code identifiers in Appendix Table 3.

Architectures and tasks. Table 1 summarizes the language-model families emphasized in the paper. Decoder-only language models use a byte-level vocabulary of 256 and context length 128. Encoder-decoder experiments use the same byte-level surface but split the sequence into a source half and a causal decoder half. TinyStories and Wikitext are the training corpora (Eldan and Li, 2023; Merity et al., 2017).

Rationale for the testbed. The byte-level harness lets us hold the placement question fixed across model families and datasets. Tokenizer, vocabulary size, sequence packing, and loss definition are identical across the decoder-only and encoder-decoder sweeps, so the only thing that changes inside each matrix is where the surviving RMSNorm layers sit. This narrows external validity relative to large-scale subword pretraining, but it gives a clean internal comparison for the question we are asking.

Training protocol. All runs use AdamW with weight decay 0.01, dropout 0.1, a feed-forward expansion ratio of $4\times$, and gradient clipping at norm 1.0. Within each experiment matrix, all variants share the same data split, horizon, and optimizer schedule; we do not retune hyperparameters per variant. Decoder-only medium/large/xlarge sweeps use 900 steps. The aggressive-512 corroboration, the decoder boundary probes, and the matched-μP high-scale sweeps all use a 480-step schedule, with batch size 2 at 200M–309M and width-aware μP initialization plus hidden-layer learning-rate scaling relative to base width 256. Exact hyperparameters and variant definitions are listed in the appendix materials.

Table 1: Model families used in the main language-model study. Parameter counts are approximate RMSNorm-baseline counts derived from the experiment definitions.

Family	Setting	Depth / width	Params	Primary task
Decoder LM	medium	8×256	8.6M	TinyStories, Wikitext
Decoder LM	large	10×320	16.6M	TinyStories, Wikitext
Decoder LM	xlarge	12×384	28.6M	TinyStories, Wikitext
Decoder LM	aggressive	12×512	50.8M	TinyStories, Wikitext
Decoder LM	corroboration-150M	16×768	151.7M	TinyStories, Wikitext
Decoder LM	boundary-200M	18×832	200.1M	TinyStories, Wikitext
Decoder LM	boundary-250M	22×896	252.0M	TinyStories, Wikitext
Decoder LM	boundary-300M	24×896	309.2M	TinyStories, Wikitext
Encoder-decoder LM	medium	$6{+}6 \times 256$	14.4M	TinyStories, Wikitext
Encoder-decoder LM	large	$8{+}8 \times 320$	29.8M	TinyStories, Wikitext
Encoder-decoder LM	boundary-140M	$12{+}12 \times 576$	144.1M	TinyStories, Wikitext

Metrics. The primary metric is validation loss. Runs also record max gradient norm and activation scale drift. For the high-scale decoder recipe family, throughput is benchmarked separately with synthetic training loops matched to the 200M, 252M, and 309M shapes and measured with CUDA events at model-step granularity, so the reported numbers isolate model-side computation rather than data-pipeline overhead.

Comparison policy. Because the paper asks where a fixed sparse normalization budget should be allocated, each experiment matrix keeps tokenizer, model shape, horizon, optimizer schedule, and data split fixed across compared variants. We therefore interpret the resulting deltas as recipe-conditioned placement comparisons under a shared budget, not as per-variant best-case optima after separate retuning. This distinction matters most at the 200M–309M boundary, where the claim concerns which sparse recipe remains usable under a matched training setup rather than absolute best loss after unconstrained hyperparameter search.

Evidence tiers. We label every claim as confirmed or exploratory. Confirmed claims rely on the five-seed decoder baseline sweep, the four-seed 151.7M corroboration, the four-seed $17{\times}832$ and $18{\times}832$ boundary points, the four-seed high-scale μP and GAP-GATED SPARSE+μP sweeps, and the four-seed long-horizon and 512-token probes. Single-seed AGC, residual-ramp, standard-recipe gap-gated, and special-norm rescue probes are kept as mechanistic checks: they help rule out simple explanations and bound the claim scope, but they do not support the main recommendations on their own.

4 Sparse RMSNorm Placement in Causal Decoders

4.1 The output side is the preferred sparse anchor in causal decoders

Figure 1 summarizes the decoder-only result under the standard AdamW recipe. Output-anchored SPARSE-FINAL obtains a positive validation-loss margin over dense FULL-RMS at every multi-seed decoder scale from 8.6M up to the four-seed 151.7M corroboration. On the medium/large/xlarge sweep the per-setting margin $\ell_{\text{FULL-RMS}} - \ell_{\text{SPARSE-FINAL}}$ sits between 0.160 and 0.215 nats on Wikitext and between 0.194 and 0.212 nats on TinyStories. Placement also matters beyond simply reducing the norm count: with the same sparse budget, LATE-FINAL consistently trails SPARSE-FINAL, and FINAL-ONLY is usually worse than either. The optimization trace tells the same story qualitatively: SPARSE-FINAL keeps a small max gradient norm where FINAL-ONLY develops large gradient spikes.

The aggressive-512 corroboration confirms the ordering at one step up: SPARSE-FINAL reaches 1.906 ± 0.041 on TinyStories and 2.300 ± 0.010 on Wikitext, improving over FULL-RMS by 0.243 and 0.113 respectively (Appendix Table 6).

The four-seed 151.7M block is the strongest multi-seed point in favor of sparse placement: SPARSE-FINAL reaches 1.924 ± 0.036 versus 2.015 ± 0.038 for FULL-RMS on TinyStories and 2.305 ± 0.035 versus 2.376 ± 0.014 on Wikitext, with only modest drift change across seeds. Above 151.7M, however, the margin contracts. The four-seed $17{\times}832$ bridge is essentially at parity, the four-seed 200.1M $18{\times}832$ bridge is mildly nega-

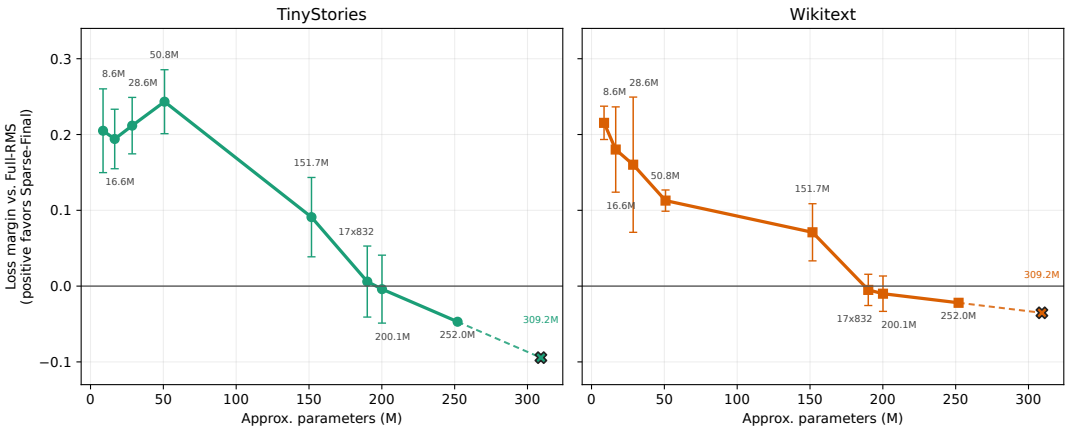

Figure 1: Decoder loss-margin curve under the standard recipe, shown separately for TinyStories and Wikitext. Each point plots the validation-loss margin $\ell_{\text{FULL-RMS}} - \ell_{\text{SPARSE-FINAL}}$ in nats; positive values favor SPARSE-FINAL. Multi-seed points extend through the four-seed 151.7M corroboration and the four-seed 17×832 and 18×832 bridge probes. The 252.0M and 309.2M markers are boundary follow-ups from two observed standard-recipe runs each; the 309.2M marker remains a failure marker rather than a fitted trend point because the sparse run diverges in both follow-ups before the 480-step budget completes.

tive on both datasets, and the higher-scale follow-ups continue the same trend: at 252.0M, dense remains ahead in both observed standard-recipe runs on both datasets (seed-7: 2.491 vs. 2.444 on TinyStories and 2.534 vs. 2.512 on Wikitext; seed-17: 2.448 vs. 2.422 and 2.487 vs. 2.457), while at 309.2M the sparse run diverges in both follow-ups before the 480-step budget completes. At the 17×832 and 18×832 points activation drift stays close to the dense baseline while the sparse max gradient norms grow sharply, so the change at the boundary is driven by gradient roughness rather than by hidden-state drift.

Two read-outs follow. First, under the standard recipe the output-anchored layout is supported only up to roughly 150M parameters: the sign change sits in the 190–250M band and the recipe fails as a drop-in replacement at 252M and 309M. Second, the evidence remains tiered across the boundary — the 151.7M block, the 17×832 bridge, and the 18×832 bridge carry four seeds each, while the 252.0M and 309.2M follow-ups remain lower-coverage checks — but the added 252M/309M run reproduces the same direction as the earlier marker rather than reversing it.

Two adjacent baselines rule out the simplest alternative explanations of the small-scale gain. DYT is stable but trails on loss (Zhu et al., 2025), and an internal-taper approximation in the style of recent gated-removal work (Kanavalau et al., 2026) is also weaker than SPARSE-FINAL in the matched large and xlarge decoder sweeps (Appendix Tables 14 and 15). The small-scale advantage therefore does not reduce to substituting RMSNorm with a smoother per-block transform; what helps is keeping the residual pathway intact and concentrating the surviving rescaling near the decoder output.

4.2 Encoder-decoder probes keep the sparse budget on the decoder side

The encoder-decoder family asks whether the decoder-only finding is about causal information flow or about the specific decoder-only architecture. In the 14M–30M regime, the decoder-side layout wins on every setting: SPARSE-DECODER obtains the lowest loss in all four configurations, improving over FULL-RMS by 0.111–0.198 (Appendix Table 5). Moving the same sparse budget to the encoder side is consistently worse — on large Wikitext, for example, the loss degrades from 2.202 to 2.359 and activation drift jumps from 0.015 to roughly 1.63 — and the 144M boundary probe shows that the encoder-decoder crossover sits earlier in parameter count than the decoder-only one. Under a matched sparse budget, then, the right side of the model for the surviving norms is the decoder side.

4.3 Mechanism and recipe sensitivity

The traces do not match a pure activation-drift explanation. On large Wikitext encoder-decoder runs, mean activation drift is the same for FULL-RMS (0.015), SPARSE-DECODER (0.015), and DECODER-FINAL (0.015), but mean max gradient

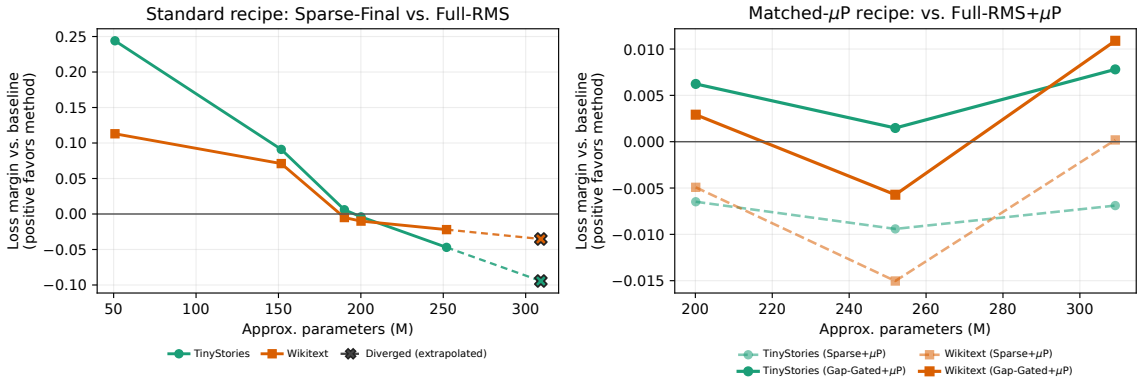

Figure 2: Recipe sensitivity at the decoder boundary. Left panel: standard-recipe loss margin against scale. The sparse advantage is positive through the four-seed 151.7M block and then disappears across the four-seed 17×832 and 200.1M 18×832 bridges; the 252.0M and 309.2M points are lower-coverage boundary follow-ups and the $\times$ symbol marks the diverged sparse 309.2M run. Right panel: under matched μP, GAP-GATED SPARSE+μP consistently improves the sparse recipe itself at 200M–309M, but the comparison to FULL-RMS+μP stays mixed across seeds.

norm is 3.37 for FULL-RMS, 4.82 for SPARSE-DECODER, and 150.15 for DECODER-FINAL (Appendix Figure 4). The wrong-side encoder anchor is qualitatively different from late-only placement: it combines elevated gradients *and* catastrophic drift, which looks like a misplaced normalization bottleneck rather than generic instability.

The picture we are left with is that sparse periodic placement works as long as each unnormalized residual gap is short enough that gradient growth stays close to the dense baseline. Late-only placement fails because the correction arrives too late, and the 200M decoder bridge shows the same pattern in sharper form: activation drift stays near 0.0014 while sparse max gradient rises from 5.20 to 14.95 on TinyStories and from 4.82 to 10.72 on Wikitext. Schedule-level stabilizers do not recover the loss lead at the boundary. At 200M, AGC leaves TinyStories at 2.382 for FULL-RMS against 2.384 for the matched sparse bridge variant and Wikitext at 2.512 against 2.516; residual ramp and longer bridge make both datasets worse.

Single-seed rescue probes. A single-seed gap-gating probe asks whether suppressing the unnormalized-gap instability is enough on its own. It is not. GAP-GATED SPARSE reduces the high-scale gradient spikes and removes the 309M TinyStories divergence, but it trails FULL-RMS on loss at every tested point (Appendix Figure 7). The cost profile is asymmetric across scales: at 200M, where plain SPARSE-FINAL is already near parity, the gates add a small extra loss penalty; at 252M and 309M they recover most of the failing sparse gap but do not flip the comparison. A parallel probe that changes the surviving norm itself behaves the same way — sparse DyT is the strongest replacement, but by 252M and 309M no special-norm variant improves over FULL-RMS (Appendix Figure 7). Affine sparse is informative for a different reason: at 252M TinyStories it reduces max gradient norm from 2280.89 to 29.19 without improving loss, which says that suppressing the gradient envelope is not by itself sufficient to recover the loss advantage. The probes are mechanistically useful, but none of them is a successful architectural rescue inside the standard recipe.

A small learning-rate sweep does not rescue 252M sparse placement. One obvious alternative explanation is that the 252M failure is only a schedule mismatch. A follow-up three-point sweep for standard-recipe SPARSE-FINAL at 252M does not support that reading. On TinyStories, 3×10^{-5}, 4×10^{-5}, and 5×10^{-5} give losses of 2.502, 2.482, and 2.529 respectively; on Wikitext the same sweep gives 2.538, 2.517, and 2.560. The middle setting is the best among the three, but it still trails the matched dense follow-up on both datasets, and the highest learning rate produces very large gradient spikes without improving loss. Within this narrow local sweep, then, the 252M boundary is not explained by simply needing a smaller learning rate.

The boundary shifts under a width-rescaled optimizer. We re-run the four high-scale settings under a matched four-seed μP protocol (Yang et al., 2021), primarily because the standard-recipe sparse 309.2M point diverges and a comparison there is otherwise impossible. With μP, SPARSE-FINAL no longer diverges at 309.2M, and the sparse loss sits within 0.015 of FULL-RMS across all six 200M–309M dataset-scale points (Appendix Table 11). The mean max gradient is modestly larger under sparse+μP — about 35–37 versus 29–32 for FULL-RMS+μP — but the catastrophic standard-recipe failure mode is gone.

Gap stabilization improves sparse+μP but not the dense comparison. Layering gap gating on top of sparse+μP lowers the mean max gradient below both SPARSE-FINAL+μP and FULL-RMS+μP at every high-scale point (Figure 2; Appendix Figure 6). The paired-seed comparison sharpens the picture: GAP-GATED SPARSE+μP beats plain SPARSE-FINAL+μP in all 24 paired runs with an overall paired delta of -0.0110 ± 0.0040, while the comparison against FULL-RMS+μP is much weaker at -0.0039 ± 0.0114 over the same 24 paired runs (Appendix Table 12). Gap gating fixes the sparse recipe but does not overturn dense normalization.

Longer training and longer context pull the result toward dense. We continue the four-seed 200M μP runs from 480 to 3840 steps for all three recipes, logging losses at the 480, 960, 1920, and 3840 milestones. Figure 3 re-indexes those milestones against the 200M model-step timings from Table 2. The pattern is the same on both datasets: GAP-GATED SPARSE+μP leads or ties at the original short budget and remains close through the intermediate milestones, but by 3840 steps dense FULL-RMS+μP is in front. On TinyStories the two recipes move from 2.559 versus 2.566 at 480 to 1.821 versus 1.770 at 3840, and on Wikitext from 2.603 versus 2.606 at 480 to 2.202 versus 2.179 at 3840. A separate four-seed 200M probe at 512-token context tells the same story from a different direction: TinyStories stays near parity, while Wikitext settles back to FULL-RMS+μP at 2.542 ± 0.033, against 2.562 ± 0.042 for GAP-GATED SPARSE+μP and 2.572 ± 0.046 for SPARSE-FINAL+μP.

A capacity-versus-conditioning sketch consistent with the observed sign change. We do not derive the crossover, but a simple post-hoc sketch is consistent with where it sits. Removing a surviving RMSNorm restores one radial degree of freedom out of d, giving a representational term of order $1/d$; the conditioning cost is taken to accumulate over the U unnormalized residual gaps, with each gap contributing a Jacobian factor $I+G_u$ whose spectral excess is taken to grow with width as $\|G_u\|_2 \propto \sqrt{d}$. Putting these together gives a signed margin

$$\Delta(L,d) \approx \frac{a}{d} - b\,U\sqrt{d}, \qquad U \approx \frac{L}{2},$$

and a crossover score $S(L,d) = Ud^{3/2}$. We stress that this is a fitted summary rather than a derivation: with two or three positive anchor points we cannot separate $Ud^{3/2}$ from neighbouring functional forms such as Ud or $\sqrt{U}\,d$ (Appendix B). What it does buy is a single scalar that the empirical sign changes are compatible with, with the fitted zero of Δ sitting inside the observed 200M–250M boundary band.

4.4 Efficiency benefits remain local to the short-budget envelope

Table 2 reports modest CUDA-event model-step gains for GAP-GATED SPARSE+μP over FULL-RMS+μP: +6.7% at 200M, +3.9% at 252M, and +6.4% at 309M. Figure 3 maps the same advantage onto wall-clock minutes: the throughput edge keeps GAP-GATED SPARSE+μP competitive at 200M for the short budget, but dense FULL-RMS+μP continues to improve faster as training extends. Appendix Table 13 broadens the comparison to all tested norm variants: faster variants such as sparse DyT do not sit on the loss frontier. The throughput advantage of gap-gated sparse is real but small, and it matters only inside the part of the operating envelope where the loss gap is already small.

4.5 Takeaways

Read together, our experiments support a scope-bounded recommendation rather than a general sparse-normalization rule. Below roughly 150M parameters and under the standard AdamW recipe, the strongest configuration we observed in this harness is output-anchored SPARSE-FINAL; in encoder-decoder models, the same sparse budget should sit on the decoder side rather than the encoder side.

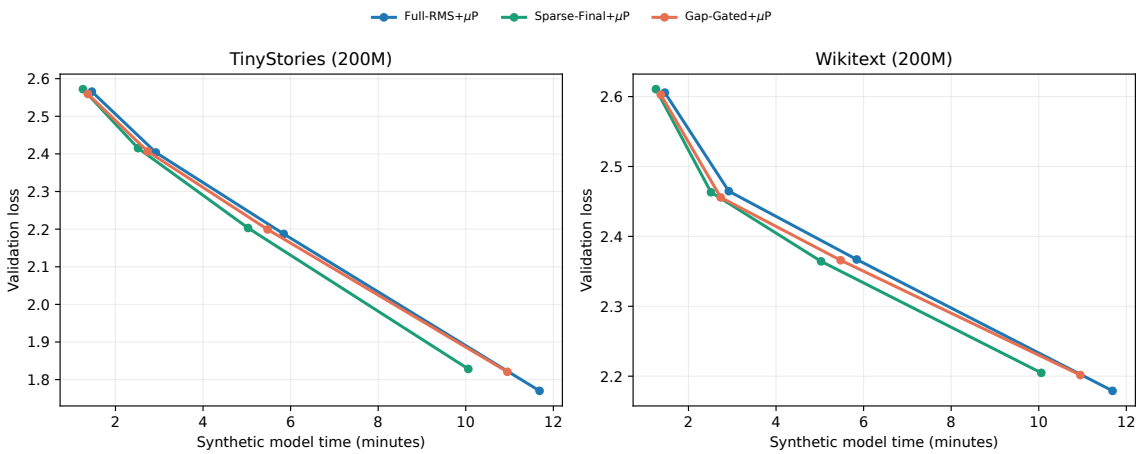

Figure 3: Compute-matched 200M frontier. Validation loss is plotted against synthetic model time by re-indexing the 200M horizon milestones with the 200M model-step timings from Table 2. GAP-GATED SPARSE+μP is competitive at the original short budget, but dense FULL-RMS+μP pulls ahead as the horizon extends.

Table 2: CUDA-event model-step throughput for the main high-scale decoder recipes under matched μP. Throughput is measured on synthetic loops that match the 200M, 252M, and 309M decoder shapes and isolate model-side computation. Entries are mean steps/s $\pm$ standard deviation over repeated runs; percentages are relative to FULL-RMS+μP at the same scale.

Method	200M	252M	309M
FULL-RMS+μP	5.48±0.57	5.03±0.62	4.84±0.50
SPARSE-FINAL+μP	6.36±0.97 (+16.2%)	5.54±0.32 (+10.2%)	5.19±0.46 (+7.3%)
GAP-GATED SPARSE+μP	5.85±0.84 (+6.7%)	5.23±0.56 (+3.9%)	5.15±0.17 (+6.4%)

In the 200M–309M range, the right sparse recipe is GAP-GATED SPARSE+μP: it beats SPARSE-FINAL+μP in every paired run, keeps the max gradient envelope below either FULL-RMS+μP or SPARSE-FINAL+μP, and retains a small throughput lead. It still does not replace dense normalization. Across the 24 paired high-scale runs the gap-versus-dense paired delta is -0.0039 ± 0.0114, with a 95% confidence interval that crosses zero, while the gap-versus-sparse paired delta is -0.0110 ± 0.0040, with a 95% confidence interval that does not (Appendix Table 12). The honest reading at high scale is parity-with-stability, not dominance.

5 Conclusion

Inside this byte-level LM harness, output-anchored sparse RMSNorm is a real but bounded improvement. Below roughly 150M parameters and under the standard AdamW recipe, SPARSE-FINAL consistently lowers validation loss against dense FULL-RMS, and the same sparse budget transfers to the decoder side rather than the encoder side in encoder-decoder models. From 200M upward, the picture changes: μP removes the catastrophic standard-recipe failure and gap gating tightens the sparse μP envelope further, but dense normalization stays competitive and pulls ahead under longer training or longer Wikitext context.

For small causal decoders trained on short budgets, output-anchored sparse periodic RMSNorm is a defensible choice in this testbed. Past that range, we read our results as a boundary study: the sparse recipe can be stabilized at higher scale, but the evidence supports parity-with-stability and small efficiency gains, not a replacement for dense normalization. The fitted score $S(L, d) = Ud^{3/2}$ is a summary of the observed sign change, not a derived law; the throughput numbers are synthetic; and the longer-horizon and 512-token probes are explicit reminders that the boundary moves with budget and with context length.

6 Limitations

Five limitations bound the claim.

- **Scale and compute envelope.** Multi-seed evidence reaches 151.7M for decoder-only and 14M–30M for encoder-decoder, with four-seed μP and GAP-GATED SPARSE+μP sweeps at 200M–309M and only low-coverage standard-recipe follow-ups at 252M and 309M. Anything

above 309M is unverified, and every experiment fits inside a single 24GB GPU.

- **Dataset and tokenization regime.** The main language-modelling evidence comes from TinyStories and Wikitext with a byte-level vocabulary of 256 and context length 128; we have not run a subword-tokenization sanity check.
- **Adjacent baselines.** DyT and the gated-removal-inspired internal-taper comparison are matched-harness reimplementations on the large and xlarge decoder shapes; they are not full reproductions of the original recipes.
- **Mechanism.** The $S(L, d) = Ud^{3/2}$ score is a fitted summary, not a derived law, and it does not yet make an out-of-sample prediction.
- **High-scale practical scope.** The μP and GAP-GATED SPARSE+μP sweeps are decoder-only, and the throughput numbers are CUDA-event model-step measurements on synthetic loops rather than end-to-end training runs.

The supported claim is narrow: with a fixed sparse norm budget, output-anchored sparse RMSNorm improves validation loss for small causal decoders, but the scaling evidence here should be read as a boundary study rather than a general law.

References

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. 2021. Rezero is all you need: Fast convergence at large depth. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 1352–1361.

Andy Brock, Soham De, Samuel L. Smith, and Karen Simonyan. 2021. High-performance large-scale image recognition without normalization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1059–1071.

Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*.

Stefan Heimersheim. 2024. You can remove GPT-2's LayerNorm by fine-tuning. *arXiv preprint arXiv:2409.13710*.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. 2020. Improving transformer optimization through better initialization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4475–4483.

Andrei Kanavalau, Carmen Amo Alonso, and Sanjay Lall. 2026. Gated removal of normalization in transformers enables stable training and efficient inference. *arXiv preprint arXiv:2602.10408*.

Yohan Kim, Yao Fu, Jaehoon Lee, and Jaewoo Kang. 2025. Peri-ln: Revisiting normalization layer in the transformer architecture. In *International Conference on Machine Learning*. ArXiv preprint arXiv:2502.02732.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. 2020. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5747–5763.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong. Association for Computational Linguistics.

Sam Shleifer and Myle Ott. 2022. Normformer: Improved transformer pretraining with extra normalization. In *International Conference on Learning Representations*. ArXiv preprint arXiv:2110.09456.

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

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. 2021. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, volume 34, pages 17084–17097.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. 2019. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*.

Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, and Zhuang Liu. 2025. Transformers without normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14901–14911.

A Variant Mapping For Reproducibility

This section maps the paper-facing names used in the narrative to the exact experiment-config and result-directory identifiers used in the project artifacts. The mapping keeps the main text readable while preserving one-to-one reproducibility.

Table 3: Mapping from paper terminology to experiment-config/result identifiers.

Paper name	Config/result identifier
DYT	`dyt`
FULL-RMS	`rmsnorm`
FINAL-ONLY	`final_norm_only`
LATE-FINAL	`last2_norm_final`
SPARSE-FINAL	`every_other_norm_final`
GAP-GATED SPARSE	`every_other_norm_final_gap_rezero`
DECODER-FINAL	`decoder_final_norm_only`
LATE-DECODER	`last2_norm_decoder_final`
SPARSE-DECODER	`every_other_norm_decoder_final`
EMBEDDING-ONLY	`encoder_embedding_norm_only`
SPARSE-ENCODER	`every_other_norm_encoder_embedding`
ViT embedding-side sparse	`every_other_norm_embedding`

B Capacity-Versus-Conditioning Sketch

This appendix records the post-hoc capacity/conditioning sketch used in Section 4.3. The sketch is not a theorem; it is an explicit functional form that fits the observed sign change with two parameters, which lets us check whether the boundary band is at least consistent with a simple capacity-versus-conditioning tradeoff.

Let d be model width and U the number of unnormalized residual gaps created by sparse periodic placement. In the decoder-only SPARSE-FINAL recipe, $U \approx L/2$. Each RMSNorm removes one radial degree of freedom, so removing a fraction of those norms restores a relative representational term of order $1/d$. We write that as

$$B_{\text{cap}}(d) = \frac{a}{d},$$

where a absorbs task-dependent constants.

The conditioning cost accumulates across the unnormalized gaps. Writing the backward Jacobian of one such gap as $J_u = I + G_u$, a first-order bound gives

$$\log \kappa_{\text{sparse}} - \log \kappa_{\text{rms}} \approx \sum_{u=1}^{U} \log(1 + \|G_u\|_2) \approx U\,\bar{\epsilon}(d).$$

The recorded traces show that gradient amplification grows with width; the simplest width law consistent with that trend is $\bar{\epsilon}(d) \propto \sqrt{d}$. This yields

$$C_{\text{cond}}(L, d) = b\,U\sqrt{d},$$

and therefore the signed sparse-margin diagnostic

$$\Delta(L, d) \approx \frac{a}{d} - b\,U\sqrt{d}.$$

The crossover is the zero of Δ, which is equivalent to a threshold on

$$S(L, d) = U d^{3/2}.$$

The score gives a simple consistency check. Define the empirical decoder margin as the mean of $\ell_{\text{rms}} - \ell_{\text{sparse}}$ across TinyStories and Wikitext, so positive values favor SPARSE-FINAL. For the clearly positive decoder points, the aggressive-512 block gives $m = 0.178$ at $S = 6 \cdot 512^{3/2} \approx 6.95 \times 10^4$, and

the 4-seed 151.7M corroboration gives $m = 0.083$ at $S = 8 \cdot 768^{3/2} \approx 1.70 \times 10^5$. A two-point linear fit $m(S) = \alpha - \beta S$ through these positive points has its zero at $S^\star \approx 2.86 \times 10^5$, which lies between the measured 200.1M 18×832 probe ($S \approx 2.16 \times 10^5$, slightly negative) and the 252.0M 22×896 probe ($S \approx 2.95 \times 10^5$, clearly negative).

If we instead use the near-zero 4-seed 17×832 block ($m = 0.002$, $S \approx 2.04 \times 10^5$) to anchor the fit, the zero shifts to $S^\star \approx 2.04 \times 10^5$, which lies between the measured 17×832 parity and 18×832 negative points. Both fits therefore place the diagnostic crossover in the same 200M–250M band in which the empirical sign changes. We stress that this is a consistency check rather than an independent prediction; it is fit on two or three points and does not have enough degrees of freedom to distinguish the proposed $U\sqrt{d}$ conditioning cost from nearby alternatives such as Ud or $\sqrt{U}\,d$. The diagnostic therefore supports compatibility with a simple capacity-versus-conditioning interpretation, not uniqueness of that interpretation.

C Detailed Language-Model Tables

This appendix collects the exact decoder-only and encoder-decoder language-model numbers behind the main figures and text. We group tables by claim rather than by run order: the first subsection covers the baseline low-to-mid-scale sweeps, the second localizes the standard-recipe boundary, and the third covers the high-scale rescues and their throughput tradeoffs.

C.1 Baseline Sweeps and Corroboration

The tables in this subsection back the small-scale positive result. They show that output-anchored sparse placement obtains the lowest loss on the base decoder sweep, stays strongest in the 50.8M aggressive corroboration, and is still positive in the 151.7M four-seed block at the edge of the boundary band.

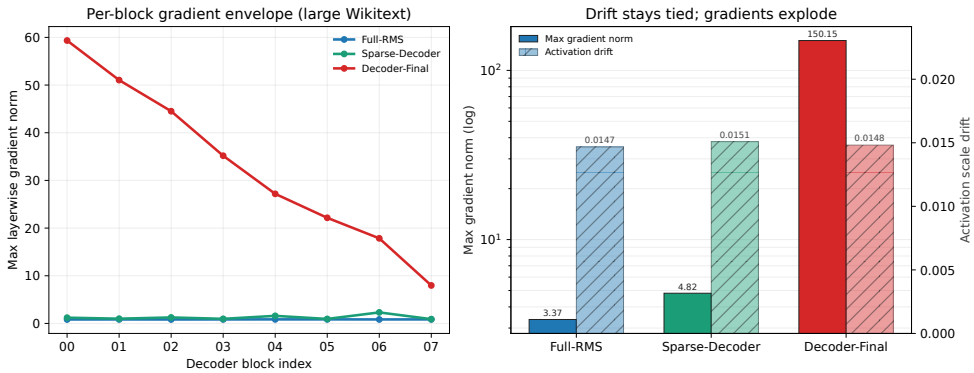

Figure 4: Mechanism evidence from the large Wikitext encoder-decoder sweep. SPARSE-DECODER stays near the FULL-RMS gradient envelope, while DECODER-FINAL exhibits extreme layerwise gradient spikes without additional activation drift.

Table 4: Decoder-only language-model sweep. Entries are mean validation loss from the five-seed aggregate used in the main paper figure. Lower is better.

Setting	SPARSE-FINAL	LATE-FINAL	FINAL-ONLY	FULL-RMS
TinyStories-medium	**1.442**	1.478	1.534	1.647
Wikitext-medium	**1.986**	2.023	2.058	2.201
TinyStories-large	**1.553**	1.664	1.796	1.747
Wikitext-large	**2.095**	2.182	2.243	2.276
TinyStories-xlarge	**1.698**	1.906	2.019	1.910
Wikitext-xlarge	**2.236**	2.337	2.404	2.396

Table 5: Encoder-decoder language-model sweep. Entries are mean validation loss ± standard deviation. Lower is better.

Setting	SPARSE-DECODER	LATE-DECODER	DECODER-FINAL	FULL-RMS
TinyStories-medium	**1.708±0.059**	1.723±0.058	1.772±0.062	1.892±0.064
Wikitext-medium	**2.180±0.055**	2.186±0.048	2.191±0.046	2.292±0.052
TinyStories-large	**1.721±0.045**	1.795±0.042	1.894±0.037	1.919±0.035
Wikitext-large	**2.202±0.085**	2.254±0.073	2.278±0.078	2.313±0.081

Table 6: Aggressive-512 decoder-only corroboration. Entries are mean ± standard deviation over three seeds. Lower is better for loss; lower max gradient indicates smoother optimization.

Setting	SPARSE-FINAL	LATE-FINAL	FULL-RMS / FINAL-ONLY
TinyStories loss	**1.906±0.041**	2.037±0.021	2.150±0.012 / 2.149±0.013
TinyStories max-grad	3.777±0.203	6.918±1.244	3.035±0.016 / 129.988±124.286
Wikitext loss	**2.300±0.010**	2.371±0.014	2.413±0.010 / 2.409±0.006
Wikitext max-grad	3.754±0.114	7.145±2.952	2.953±0.029 / 56.758±60.282

Table 7: Higher-scale decoder-only corroboration and upper boundary evidence. The 151.7M block reports four-seed mean validation loss ± standard deviation. The 309.2M sparse recipe diverges in both observed standard-recipe follow-ups, so it is reported separately as a failure marker rather than as a successful scale extension. Here ‘NaN‘ means the validation loss became undefined before the scheduled 480 steps completed: the seed-7 TinyStories and Wikitext runs fail at steps 297 and 359, while the seed-17 follow-up fails earlier at steps 240 and 220.

Setting	SPARSE-FINAL	FULL-RMS	Sparse max-grad	Sparse drift
TinyStories-150M loss	**1.924±0.036**	2.015±0.038	5.352±0.166	0.0032±0.0001
Wikitext-150M loss	**2.305±0.035**	2.376±0.014	5.632±0.459	0.0030±0.0001
TinyStories-300M loss	NaN	2.410	34493.520	0.015
Wikitext-300M loss	NaN	2.492	4589.857	0.111

Table 8: Decoder boundary markers used to localize the standard-recipe crossover. The 151.7M and 17×832 rows are four-seed confirmation points, the 200.1M 18×832 row is a four-seed bridge, and the 252.0M / 309.2M rows report the two observed standard-recipe follow-ups. For the two-run rows, losses are shown as seed-7 / seed-17 and the final column summarizes sparse drift or failure. Lower loss is better.

Setting	Seeds	SPARSE-FINAL	FULL-RMS	Sparse max-grad	Drift / failure
TinyStories-151.7M	4	**1.924±0.036**	2.015±0.038	5.352±0.166	drift 0.0032 ± 0.0001
Wikitext-151.7M	4	**2.305±0.035**	2.376±0.014	5.632±0.459	drift 0.0030 ± 0.0001
TinyStories-17x832	4	**2.420±0.036**	2.425±0.031	8.738±1.531	drift 0.0014 ± 0.0000
Wikitext-17x832	4	2.473±0.026	**2.472±0.032**	8.757±0.404	drift 0.0014 ± 0.0000
TinyStories-18x832	4	2.424±0.034	**2.420±0.029**	10.816±2.521	drift 0.0015 ± 0.0000
Wikitext-18x832	4	2.475±0.029	**2.467±0.031**	13.472±2.686	drift 0.0014 ± 0.0000
TinyStories-252.0M	2	2.491 / 2.448	**2.444 / 2.422**	78.628 / 17.464	no NaN; drift 0.0013 / 0.0012
Wikitext-252.0M	2	2.534 / 2.487	**2.512 / 2.457**	58.651 / 199.296	no NaN; drift 0.0012 / 0.0012
TinyStories-309.2M	2	NaN / NaN	**2.410 / 2.406**	34493.520 / 9080.642	NaN at 297 / 240
Wikitext-309.2M	2	NaN / NaN	**2.492 / 2.447**	4589.857 / 1663.460	NaN at 359 / 220

These rows make the boundary split visible at the numeric level. The 151.7M block is a four-seed positive aggregate, the 17×832 and 18×832 rows localize the crossover with four seeds each, and the 252.0M and 309.2M rows show that a second standard-recipe follow-up reproduces the same higher-scale direction rather than reversing it.

C.2 Boundary Localization and Failed Standard-Recipe Rescues

The next set of results pins down where the standard recipe stops favouring sparse placement. They also document schedule-level rescue attempts that improved optimization smoothness without restoring the sparse loss lead; this is why we treat them as boundary-localization evidence in the main text rather than as recommended recipes in their own right.

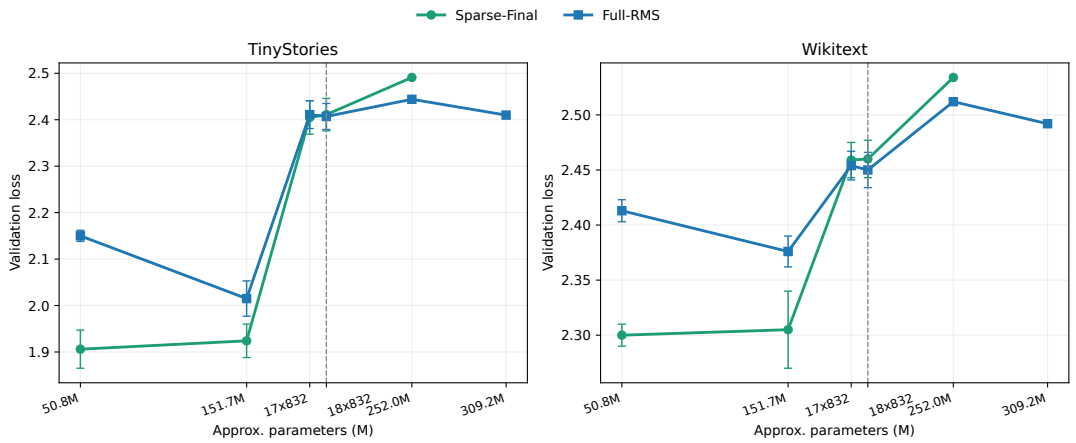

Figure 5: Decoder boundary localization in exact losses rather than only signed margins. Each point plots the completed TinyStories or Wikitext decoder runs from 50.8M through the nominal 309.2M boundary marker, with error bars on multi-seed points. At 309.2M, the dense FULL-RMS point is shown while the sparse result is omitted from the curve because it diverges before completion. This figure replaces the previous pair of dense decoder boundary tables while keeping the same crossover evidence visible at a glance.

Table 9: Encoder-decoder sparse-placement evidence. The first four rows are the four-seed 14M–30M sweep and expose both decoder-side and encoder-side sparse placement directly. The final rows are the single-seed 144.1M boundary probe, where decoder-side sparse placement is already negative relative to FULL-RMS. The Wikitext DECODER-FINAL 'NaN' is a true divergence marker rather than a missing evaluation: its validation loss becomes undefined at step 417.

Setting	SPARSE-DECODER	FULL-RMS	SPARSE-ENCODER	DECODER-FINAL
TinyStories-medium loss	**1.708±0.059**	1.892±0.064	1.794±0.058	1.772±0.062
Wikitext-medium loss	**2.180±0.055**	2.292±0.052	2.271±0.051	2.191±0.046
TinyStories-large loss	**1.721±0.045**	1.919±0.035	1.892±0.048	1.894±0.037
Wikitext-large loss	**2.202±0.085**	2.313±0.081	2.359±0.101	2.278±0.078
TinyStories-144.1M loss	2.604	**2.578**	2.625	2.929
TinyStories-144.1M max-grad	10.104	6.342	60.079	33472.328
TinyStories-144.1M drift	0.0021	0.0021	1.8971	0.0039
Wikitext-144.1M loss	2.581	**2.573**	2.586	NaN
Wikitext-144.1M max-grad	10.568	6.489	64.123	15117.975
Wikitext-144.1M drift	0.0020	0.0019	1.8429	0.0305

Table 10: Failed high-scale stabilization rescues. Entries are single-seed validation loss after adding fair schedule-level and optimizer-level stabilizers at the first failing boundary points. AGC matches or improves gradient envelopes, but does not restore the sparse loss advantage. Residual ramp and longer bridge both worsen performance further.

Setting	FULL-RMS+AGC	bridge+AGC	bridge+ramp+AGC	long bridge+AGC
Decoder 200M TinyStories	**2.382**	2.384	2.403	2.411
Decoder 200M Wikitext	**2.512**	2.516	2.528	2.530
Enc-Dec 140M TinyStories	**2.579**	2.598	2.596	2.627
Enc-Dec 140M Wikitext	**2.562**	2.573	2.582	2.606

C.3 High-Scale Recipe Comparisons and Throughput

This subsection backs the high-scale section of the main text. The intended reading order is: μP removes the catastrophic standard-recipe failure, gap gating then improves the sparse+μP recipe itself and further suppresses gradients, the comparison to FULL-RMS+μP remains close rather than decisive, and the model-side throughput gains are measurable but small.

Table 11: Matched four-seed μP probe at the high-scale decoder boundary points. Entries are mean validation loss $\pm$ standard deviation under a μP-rescaled learning-rate and initialization recipe, together with mean max gradient norm. μP prevents the standard-recipe 309.2M divergence and keeps sparse loss within a small gap of FULL-RMS at all three high-scale points.

Setting	FULL-RMS+μP loss	SPARSE-FINAL+μP loss	Sparse − FULL-RMS	FULL-RMS max grad	SPARSE-FINAL max grad
TinyStories-200M	**2.566±0.042**	2.572±0.044	0.006	29.32	37.22
Wikitext-200M	**2.606±0.077**	2.611±0.078	0.005	29.20	36.92
TinyStories-252M	**2.547±0.028**	2.557±0.039	0.009	29.15	35.00
Wikitext-252M	**2.593±0.071**	2.608±0.071	0.015	31.00	36.86
TinyStories-309M	**2.489±0.030**	2.496±0.021	0.007	31.73	35.73
Wikitext-309M	2.531±0.054	**2.531±0.063**	-0.0002	31.81	35.60

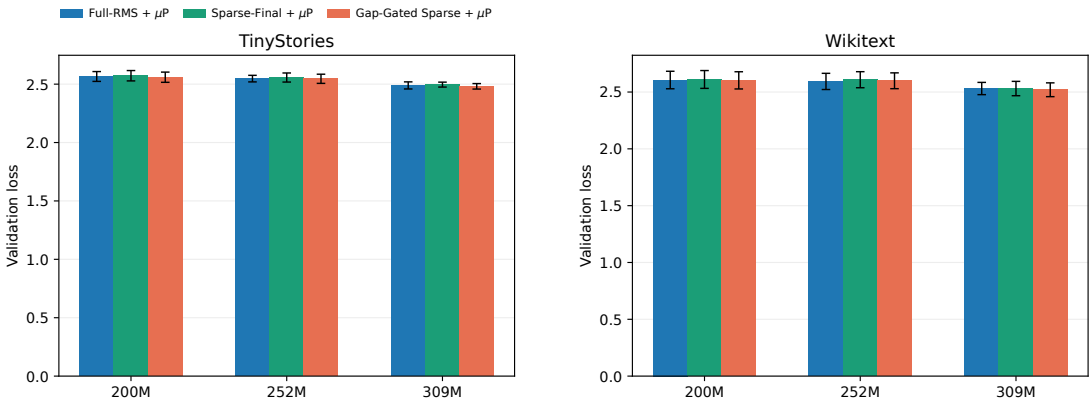

Figure 6: Four-seed GAP-GATED SPARSE+μP confirmation at the high-scale decoder points. The grouped bars show mean validation loss with standard-deviation error bars for FULL-RMS+μP, SPARSE-FINAL+μP, and GAP-GATED SPARSE+μP at 200M, 252M, and 309M on both datasets. This figure replaces the previous two-row-per-setting table while keeping the uncertainty visible.

The next figure compresses the single-seed standard-recipe rescue block into one heatmap. It is intended as mechanistic evidence rather than as a new headline result: gap gating stabilizes the high-scale sparse recipe and sparse-DyT is the strongest among the special-norm replacements, but no architectural variant in this block establishes a consistent improvement over FULL-RMS.

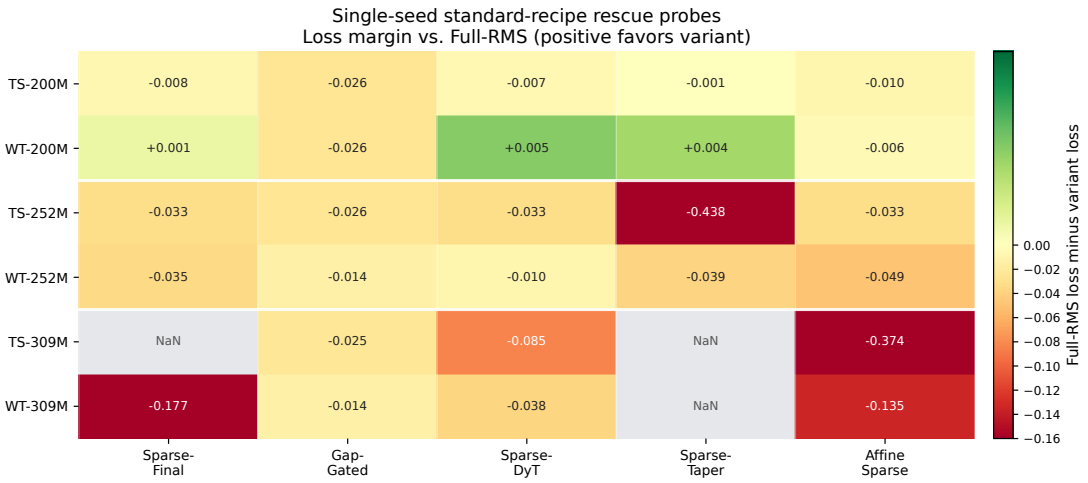

Figure 7: Single-seed standard-recipe rescue summary at the 200M–309M decoder boundary. Each cell reports the exact loss margin $\ell_{\text{FULL-RMS}} - \ell_{\text{variant}}$; green cells favor the rescue variant, red cells favor FULL-RMS, and gray cells denote true divergences before the 480-step budget completed. In this block, the gray cells correspond to SPARSE-FINAL on TinyStories-309M (diverges at step 453) and sparse-TaperNorm on TinyStories-309M and Wikitext-309M (diverge at steps 401 and 324 respectively). The pattern is visually consistent with the main text: gap gating recovers much of the failing sparse loss gap without turning the comparison positive, sparse-DyT is the strongest special-norm replacement, and no architectural rescue re-establishes a clear FULL-RMS replacement in the standard recipe.

Table 12: Paired-seed deltas for the four-seed high-scale GAP-GATED SPARSE+μP confirmation. Negative values favor GAP-GATED SPARSE+μP. The robust result is that GAP-GATED SPARSE+μP improves over plain SPARSE-FINAL+μP in every paired run; comparison to FULL-RMS+μP is favorable overall but mixed across seeds.

Setting	n	Gap − Full	Signs	Gap − Sparse
TinyStories-200M	4	−0.0062±0.0101	3−/1+	−0.0127±0.0013
Wikitext-200M	4	−0.0029±0.0077	3−/1+	−0.0078±0.0030
TinyStories-252M	4	−0.0015±0.0112	2−/2+	−0.0109±0.0023
Wikitext-252M	4	+0.0057±0.0081	1−/3+	−0.0093±0.0014
TinyStories-309M	4	−0.0078±0.0119	3−/1+	−0.0147±0.0034
Wikitext-309M	4	−0.0109±0.0168	3−/1+	−0.0107±0.0072
Overall	24	−0.0039±0.0114	15−/9+	−0.0110±0.0040

Table 13: Full-family clean CUDA-event model-step throughput at the high-scale decoder boundary shapes under matched μP. Values are mean steps/s ± standard deviation; percentages in parentheses are relative to FULL-RMS+μP at the same scale. Faster variants are not necessarily better recipes: sparse-DyT is fast, but Appendix Figure 7 shows that it still loses on validation loss.

Method	200M	252M	309M
FULL-RMS+μP	5.48±0.57	5.03±0.62	4.84±0.50
SPARSE-FINAL+μP	6.36±0.97 (+16.2%)	5.54±0.32 (+10.2%)	5.19±0.46 (+7.3%)
GAP-GATED SPARSE+μP	5.85±0.84 (+6.7%)	5.23±0.56 (+3.9%)	5.15±0.17 (+6.4%)
sparse-DyT	6.80±1.67 (+24.2%)	6.03±0.78 (+19.9%)	5.70±0.37 (+17.9%)
sparse-TaperNorm	4.21±0.56 (−23.2%)	3.59±0.43 (−28.5%)	3.78±0.33 (−21.8%)
affine sparse	6.24±1.31 (+14.0%)	5.54±0.53 (+10.1%)	5.31±0.59 (+9.8%)

C.4 Adjacent Baselines

The final tables in this appendix document the two neighbouring normalization baselines discussed briefly in the main text. They show that the small-scale positive result is not just an artefact of swapping RMSNorm for a smoother per-block transform: in the matched large and xlarge decoder sweeps, both adjacent baselines are weaker than SPARSE-FINAL.

Table 14: Narrow DYT adjacent-baseline comparison on decoder-only language modeling. Entries are mean validation loss ± standard deviation over seeds ‘7‘ and ‘17‘. Lower is better.

Setting	DYT	SPARSE-FINAL	LATE-FINAL	FINAL-ONLY	FULL-RMS
TinyStories-large	2.115±0.006	**1.585±0.023**	1.688±0.019	1.805±0.031	1.763±0.013
Wikitext-large	2.465±0.067	**2.118±0.074**	2.195±0.081	2.253±0.079	2.283±0.044
TinyStories-xlarge	2.228±0.015	**1.724±0.016**	1.918±0.0004	2.036±0.003	1.923±0.011
Wikitext-xlarge	2.503±0.101	**2.269±0.110**	2.383±0.117	2.448±0.125	2.426±0.104

Table 15: Narrow gated-removal-inspired adjacent-baseline comparison on decoder-only language modeling. Entries are mean validation loss ± standard deviation over seeds ‘7‘ and ‘17‘. Lower is better.

Setting	gated-removal-inspired	SPARSE-FINAL	FULL-RMS
TinyStories-large	1.620±0.023	**1.585±0.023**	1.763±0.013
Wikitext-large	2.201±0.064	**2.118±0.074**	2.283±0.044
TinyStories-xlarge	1.765±0.010	**1.724±0.016**	1.923±0.011
Wikitext-xlarge	2.422±0.097	**2.269±0.110**	2.426±0.104

D Compute and Runtime Details

All reported experiments ran in the project environment on a single NVIDIA GPU with roughly 24GB of memory. Experiment matrices were launched through versioned YAML configs and resumable scripts rather than ad-hoc notebook runs, and we keep completed, interrupted, and failed runs in the record. This matters here because several appendix rows are explicit boundary or failure markers rather than only successful completions: the negative result is part of the numeric record, not a post-hoc omission.

E Reproducibility, Statistics, and Responsibility

We label every claim as confirmed or exploratory. Confirmed results include the five-seed low-scale decoder sweep, the four-seed 151.7M corroboration, the four-seed 17×832 and 18×832 bridge points, the four-seed high-scale μP and GAP-GATED SPARSE+μP sweeps, and the four-seed long-horizon and 512-token probes. Single-seed AGC, residual-ramp, standard-recipe gap-gated, and special-norm rescue runs are kept as mechanistic or boundary evidence, and are flagged that way in both the text and the captions.

We provide an anonymous artifact archive for this submission. It contains the experiment code, YAML matrices, stage launchers, figure-generation scripts, and paper source.

Main tables and figures report mean validation loss with standard deviations over seeds; the paired high-scale GAP-GATED SPARSE+μP comparison additionally reports paired deltas, sign counts, and dispersion for confidence-interval reading in Appendix Table 12. The high-scale result is still negative overall: gap gating reliably improves over plain sparse+μP, but the paired comparison against dense FULL-RMS+μP is too small to support a replacement claim.

The main responsibility concerns here are compute use, scope, and reproducibility rather than privacy or deployment harm. We address them by keeping divergences and failed runs in the record and by avoiding claims that extend beyond the verified byte-level short-budget regime.