LAYERLOOP: STAGE GRANULARITY IN RECURRENT-DEPTH LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

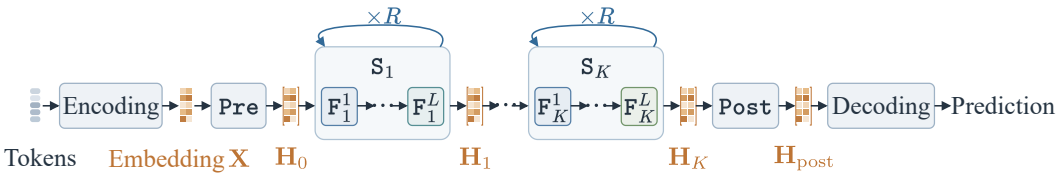
Recurrent-depth Transformers reuse stored blocks to increase executed depth, but the number of blocks repeated together is another architectural choice. We introduce LayerLoop, which divides the recurrent middle into consecutive multi-block stages and repeats each stage before advancing. This stage-partitioned design spans the space between repeating each block before advancing (blockwise recurrence) and repeating the entire middle region as one unit (whole-middle recurrence). We train and compare stage groupings at matched stored and executed depths, repeat count, and processed tokens. In the four-repeat comparison, four-block stages score 71.2 on a 24-component diagnostic suite of reference-relative percentile ranks, versus 55.4 for blockwise and 57.5 for whole-middle recurrence. Blockwise recurrence instead has the numerically lowest measured next-token prediction loss, showing that stage granularity changes the trained model's quality profile even at fixed depths. At similar parameter storage and processed tokens, the four-block-stage model also leads the tested dense and recurrent baselines on the suite. Across two tested widths, its stored parameters participate more broadly than those of matched dense models. Without predefined roles, its stages develop distinct, repeatable responses when their repeated blocks are bypassed. LayerLoop thus turns stage granularity into an explicit design choice, with trained-model behavior that stored and executed depth counts alone do not describe.

1 INTRODUCTION

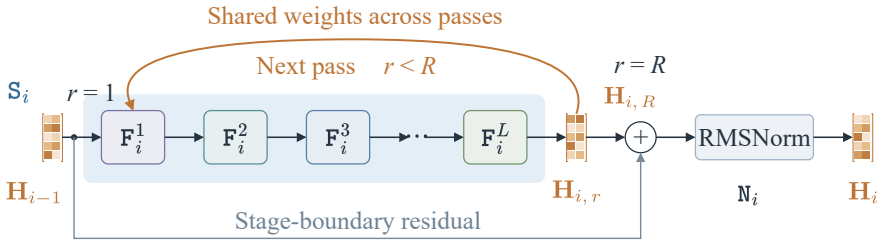
Transformers underlie modern language models (Vaswani et al., 2017). Adding depth gives each token more learned computation, but in a conventional stack every added block also introduces a new parameter set and another serial step. Recurrent-depth models apply stored blocks more than once, separating the number of distinct blocks (stored depth) from the number of serial block applications (executed depth). Existing designs reapply one shared transformation (Dehghani et al., 2018), cycle through an ordered parameter sequence (Takase & Kiyono, 2021), or repeat an entire multi-layer region (Bae et al., 2025a; Zhu et al., 2025; Geiping et al., 2025). These designs establish ways to share blocks but leave open how the number of consecutive blocks repeated together affects a trained model when storage and serial execution are held fixed. We therefore ask: *how does stage granularity affect the trained model when stored and executed depths are fixed?*

We introduce *LayerLoop* (Figure 1), a simple recurrent-depth architecture that partitions the recurrent portion into consecutive multi-block stages, repeats each complete stage, and then advances to the next. We call the number of blocks per stage *stage granularity*. Larger stages apply more distinct blocks before the next repeat begins; smaller stages begin the next repeat sooner. Intermediate stages let the model revisit a local group before advancing through the remaining distinct blocks, between repeating individual blocks and repeating the whole middle region.

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.



(a) Overall computation



(b) Recurrent stage

Figure 1: LayerLoop from input tokens to prediction. The top path advances through K consecutive stages, where K is the number of stages, between non-recurrent prefix and suffix blocks. The lower view expands stage $\mathtt{S}_i$, indexed by i: L is its number of stored blocks, and R is the number of times those blocks are applied with the same weights. Stage i receives $\mathbf{H}_{i-1}$, repeat r produces $\mathbf{H}_{i,r}$, and the stage-boundary residual and root-mean-square normalization $\mathtt{N}_i$ produce $\mathbf{H}_i$. Orange grids depict evolving representations schematically, while the differently colored block boxes distinguish stored blocks. Encoding produces the embedded token tensor $\mathbf{X}$, Pre maps $\mathbf{X}$ to $\mathbf{H}_0$, Post maps $\mathbf{H}_K$ to $\mathbf{H}_{\text{post}}$, and Decoding maps $\mathbf{H}_{\text{post}}$ to next-token predictions. For clarity, the schematic omits the repeat-conditioning vectors and residual-scale parameters defined in Section 3.

We test whether stage granularity changes the trained model using held-out next-token prediction loss and a 24-component diagnostic suite spanning language modeling, context use, and reasoning. Within each controlled comparison, the models have matched stored and executed depths, repeat counts, and processed-token budgets. Two-block stages lead both measures in the shallower two-repeat setting. With four repeats, four-block stages lead the suite, whereas blockwise recurrence has the numerically lowest prediction-loss estimate. At the same stored and executed depths, the two measures therefore select different trained stage groupings (Section 4.2).

The four-block-stage checkpoint also leads the tested dense and recurrent alternatives on the diagnostic suite at approximately matched parameter storage and processed tokens. Its suite advantage over Dense appears across five stored depths and in a wider model, so the finding is not confined to a single model size (Sections 4.3 and 4.4).

We also examine two properties of the trained architecture. Across two tested widths, LayerLoop has less concentrated first-order loss sensitivity across stored parameters and fewer rarely active multilayer-perceptron (MLP) units than storage- and token-matched dense models (Section 4.5). Separately, passing each stage's input onward instead of running its repeated blocks produces distinct patterns of task responses that recur across evaluation samples in two trained checkpoints. Replacing one complete stage with another among four tested stages increases prediction loss in all tested replacements within the fully trained checkpoint. We call these repeatable stage-specific responses and failed replacements *functional differentiation*; the stages were not assigned roles in advance (Section 4.6).

Our contribution is a stage-partitioned recurrent architecture together with controlled evidence that its intermediate grouping changes quality rankings even at equal stored and executed depths. The trained models also show broad participation across stored parameters and distinct responses from stages that were assigned no roles in advance.

2 Related Work

Recurrent-depth methods differ in which stored blocks are repeated together and how repeated computation is allocated. LayerLoop studies the first choice by partitioning the recurrent middle into consecutive multi-block stages and completing each stage's repeats before advancing. Its controlled granularity comparisons hold stored and executed block counts fixed.

Organization of the repeated unit. Universal Transformers repeatedly apply one shared transformation (Dehghani et al., 2018), while cyclic sharing reuses an ordered sequence of parameter sets across depth (Takase & Kiyono, 2021). Recent recurrent-depth language models repeat larger multi-block regions. Relaxed Recursive Transformers repeat a region of blocks with distinct stored parameters (Bae et al., 2025a); Huginn repeats a multi-layer recurrent block (Geiping et al., 2025); and Hyperloop repeats its multi-block middle region as one unit between non-recurrent prefix and suffix blocks (Zeitoun et al., 2026). Ouro also repeats a multi-layer block while learning how much recurrent computation to use (Zhu et al., 2025). These methods vary the repeated unit from one transformation to an ordered sequence or a larger region. Cyclic sharing revisits the sequence, whereas LayerLoop finishes each stage's repeats before advancing. The methods reviewed here do not compare how many consecutive stored blocks are repeated together across several ordered stages while holding stored and executed depth fixed. LayerLoop tests that intermediate grouping, rather than recurrence itself.

Execution allocation and the cost of sharing. Beyond the repeated unit, recurrent models choose how much computation to apply and whether shared blocks can substitute for independently parameterized depth. Mixture-of-Recursions routes tokens to token-specific recurrent depths (Bae et al., 2025b). Ouro learns how much recurrent depth to allocate through its training objective (Zhu et al., 2025), while LoopFormer trains one model to operate under different execution budgets (Jeddi et al., 2026). Parcae studies training stability and scaling as the repeat count changes (Prairie et al., 2026). Iso-depth scaling instead compares shared blocks with independently parameterized blocks at the same executed depth, showing that another application of shared weights does not recover the full capacity of another independently parameterized block (Schwethelm et al., 2026). These works study execution allocation or the capacity trade-off between shared and independently parameterized depth. LayerLoop instead varies which consecutive blocks form each repeated stage; its trained-model comparisons fix the repeat count, stored blocks, and total block applications.

3 LayerLoop

A decoder-only Transformer applies an ordered sequence of blocks to token representations and maps the final hidden state to next-token distributions (Vaswani et al., 2017). LayerLoop keeps this input-to-output computation but replaces the middle of the block sequence with consecutive multi-block stages that are executed repeatedly. We use *repeat* for this operation throughout: the same stage is applied again with the same block weights before execution advances to the next stage. After defining the stage partition and its depth accounting, we specify repeat conditioning and the stage-boundary residual, then identify the blockwise and whole-middle endpoints used in the experiments.

Stage-partitioned repeated execution. Let an optional prefix contain D_{pre} blocks, let the repeated middle contain M_{mid} stored blocks, and let an optional suffix contain D_{post} blocks. LayerLoop partitions the middle into K consecutive stages with L blocks each, so $M_{\text{mid}} = KL$, and executes every stage R times. We call R the *repeat count*. We call L the *stage granularity*: it is the number of consecutive blocks in a stage and therefore the number executed before that stage begins its next repeat. The number of distinct stored block parameter sets is the *stored depth* $D_{\text{unique}} = D_{\text{pre}}+M_{\text{mid}}+D_{\text{post}}$. Because every middle block is applied R times, the number of serial block applications is the *executed depth* $D_{\text{exec}} = D_{\text{pre}} + RM_{\text{mid}} + D_{\text{post}}$.

Stage $i \in \{1, \ldots, K\}$ contains stored middle blocks indexed $j = (i-1)L + 1, \ldots, iL$. Figure 1 labels its ℓ-th block $\mathtt{F}_i^\ell$, where $j = (i-1)L + \ell$ for $\ell = 1, \ldots, L$. In the equations, $\mathtt{F}_j^{(s)}$ denotes the same block with residual-branch scale $s \in (-1, 1)$; each j has distinct parameters. The corresponding

stage applies its blocks in their original order,

$$\mathtt{S}_i^{(s)} = \mathtt{F}_{iL}^{(s)} \circ \cdots \circ \mathtt{F}_{(i-1)L+1}^{(s)}, \tag{1}$$

where composition is written right to left: $\mathtt{A} \circ \mathtt{B}$ applies $\mathtt{B}$ first. Each repeat instantiates the generic scale s as the indexed value $s_{i,r}$ defined below. Each block uses the same weights whenever its stage is repeated.

Repeat-conditioning vectors and the stage-boundary residual. For a batch of b sequences of length n and hidden width d, let $\mathbf{X} \in \mathbb{R}^{b\times n\times d}$ denote the embedded token representations. Let $\mathtt{Pre}$ and $\mathtt{Post}$ denote the prefix and suffix mappings, using the identity mapping when the corresponding blocks are absent. Let $\mathbf{H}_{i,r} \in \mathbb{R}^{b\times n\times d}$ denote the hidden state at repeat position $r \in \{0,\dots,R\}$ of stage i. The prefix produces $\mathbf{H}_0 = \mathtt{Pre}(\mathbf{X})$; for each stage, $\mathbf{H}_{i,0} = \mathbf{H}_{i-1}$ is its input and $\mathbf{H}_i$ will denote its stage-boundary output. For stage i and repeat $r \in \{1,\dots,R\}$, LayerLoop learns a repeat-conditioning vector $\vec{e}_{i,r} \in \mathbb{R}^d$ and a residual-scale parameter $a_{i,r} \in \mathbb{R}$. Each $a_{i,r}$ is an independently learned scalar initialized to $\alpha = 0.1$; α is a fixed hyperparameter rather than a model parameter. The vector $\vec{e}_{i,r}$ gives shared block weights a learned signal for the current repeat. During execution, the bounded scale $s_{i,r} = \tanh(a_{i,r}) \in (-1,1)$ lets each stage and repeat modulate its residual-branch update. The repeat-conditioning vector is broadcast over batch and token positions and added once, before the first block of the stage. The scale $s_{i,r}$ is applied to every attention and multilayer-perceptron residual branch in the stage without scaling their shortcut paths. If $\mathbf{H}_{i,r-1}$ is the incoming state, repeat r produces

$$\mathbf{H}_{i,r} = \mathtt{S}_i^{(s_{i,r})}(\mathbf{H}_{i,r-1} + \vec{e}_{i,r})\,. \tag{2}$$

After all R repeats, the stage-boundary residual preserves a direct path from the original stage input by adding $\mathbf{H}_{i-1}$ once, outside the repeated block sequence. The stage-boundary normalization $\mathtt{N}_i$ then produces the stage output by normalizing the post-addition sum $\mathbf{H}_{i,R} + \mathbf{H}_{i-1}$:

$$\mathbf{H}_i = \mathtt{N}_i(\mathbf{H}_{i,R} + \mathbf{H}_{i-1})\,, \qquad i = 1,\dots,K, \tag{3}$$

where $\mathtt{N}_i$ has one learned gain per hidden coordinate (Zhang & Sennrich, 2019). The stage-boundary residual is applied once per stage, after its repeated execution. After stage K, its boundary residual and normalization produce $\mathbf{H}_K$. The suffix maps $\mathbf{H}_K$ to $\mathbf{H}_{\text{post}} = \mathtt{Post}(\mathbf{H}_K) \in \mathbb{R}^{b\times n\times d}$. The final normalization and language-model head map $\mathbf{H}_{\text{post}}$ to next-token distributions. Training and evaluation use this same computation.

Stage-granularity endpoints. When $L = 1$, each middle block forms its own stage and is repeated R times before execution advances; we call this endpoint *blockwise recurrence*. When $L = M_{\text{mid}}$, the entire middle stack forms one stage and is repeated as a whole; we call this endpoint *whole-middle recurrence*. Intermediate values place several blocks in each of several ordered stages. Changing L changes the order of distinct and repeated block applications as well as the number and placement of repeat-conditioning vectors and stage-boundary residuals.

4 Experiments

These experiments test stage granularity as a design choice beyond stored and executed depth. Controlled trained-model comparisons first show how grouping changes quality at fixed depth budgets. We then compare the selected four-block-stage model with compact alternatives at similar storage and processed-token budgets and test whether its suite advantage over Dense extends across model sizes. Finally, sensitivity and activation measurements characterize participation across stored weights, while stage-bypass responses distinguish the ordered stages.

4.1 Experimental setting

We train all local models for next-token prediction on the same prepared deduplicated-Pile stream with the GPT-NeoX tokenizer (Gao et al., 2020; Biderman et al., 2023). A processed-token budget counts each input token once, regardless of how many times the model applies a stored block to it. The data preparation, training recipe, checkpoint policy, and run-specific budgets are recorded in Appendix A.

We report held-out negative log-likelihood (NLL, nats per token; lower is better) and a 24-component diagnostic-suite score. For checkpoint m, let $\pi_{m,c}$ be component c's percentile against the same frozen 58-checkpoint reference pool after orienting its raw value so higher is better. The suite score gives each component equal weight:

$$B_{\text{suite}}(m) = \frac{1}{24} \sum_{c=1}^{24} \pi_{m,c}. \tag{4}$$

Higher B_{suite} is better; its point differences are differences in reference-relative percentiles, not differences in task accuracy. Appendix B defines the components, reference pool, ranking convention, and reporting domains. NLL sampling and task counts are specified in Appendix A.

Our controlled comparison varies the stage granularity L while matching stored depth, executed depth, repeat count, parameter storage, and processed tokens. Changing L also changes the number and placement of stage-boundary operations, so the comparison tests complete trained stage constructions rather than an isolated schedule edit. The main width-608 LayerLoop checkpoint has $(K, L, R) = (16, 4, 4)$, 68 stored blocks, 260 executed block applications, 434.16 million parameters, and 4.342 billion processed tokens. The separate compact-model comparison approximately matches stored parameters and processed tokens against Dense, Huginn, Mixture-of-Recursions, Ouro, and Parcae, but not executed depth. Public Pythia and OLMo checkpoints provide contextual scale references, not controlled architecture matches. Appendix F details the recurrent implementations.

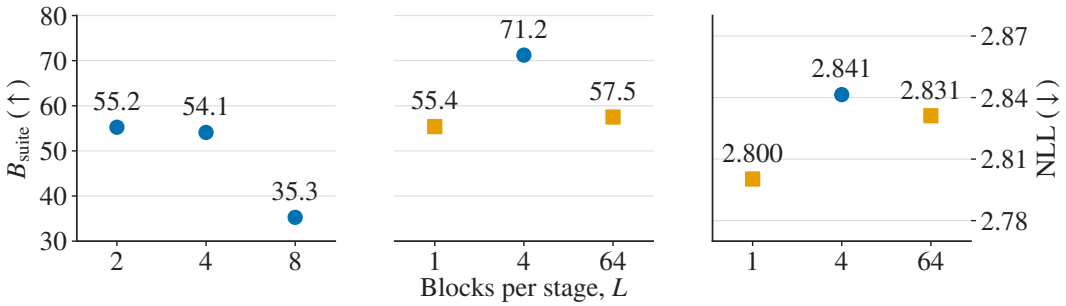
Headline comparisons use final checkpoints, with one training run per architecture–configuration pair. We also test whether the suite ordering carries over to seven established benchmark families and a separate frozen controlled-task set. Appendix F gives their scoring and paired-test protocol, aggregate results, and the supplementary paired-test record.

4.2 Stage granularity and trained-model quality at fixed depth

The central experiment asks whether stage granularity changes the trained model when storage and recurrent execution are held fixed. We use two complementary comparisons at width 608. The first varies $L \in \{2, 4, 8\}$ at $M_{\text{mid}} = 32$ and $R = 2$. The second uses the deeper main-model setting $M_{\text{mid}} = 64$ and $R = 4$ to compare $L = 4$ with the two recurrence endpoints $L = 1$ and $L = 64$; $L = 2$ was not trained in this second comparison. Within each comparison, stored depth, executed depth, repeat count, parameter count, and processed tokens are fixed or closely matched.

Stage designs reach different quality levels at fixed depth budgets. In the first comparison, we fix $D_{\text{unique}} = 36$, $D_{\text{exec}} = 68$, approximately 244.8M parameters, and 2.448B processed tokens while varying $L \in \{2, 4, 8\}$. The non-paired NLL estimates range from 2.978 at $L = 2$ to 3.494 at $L = 8$, a span of 0.516 (Appendix Figure 7a), while B_{suite} spans about 20 points (Figure 2, left). Two-block stages lead both measures, four-block stages remain close on the aggregate, and eight-block stages are markedly weaker. The trained models therefore differ across complete stage constructions even though they store and apply the same number of Transformer blocks. Because K changes with L, this comparison includes the corresponding change in stage-boundary and repeat-conditioning parameters.

Four-block stages outperform both recurrence endpoints on the suite. In the deeper comparison, the candidate set is $L \in \{1, 4, 64\}$ with $D_{\text{unique}} = 68$ and $D_{\text{exec}} = 260$. Four-block stages obtain $B_{\text{suite}} = 71.2$, ahead of whole-middle recurrence (57.5) and blockwise recurrence (55.4; Figure 2b). Blockwise recurrence instead has the numerically lowest measured NLL (Figure 2c). Thus, among these three schedules, the four-block-stage model leads the diagnostic suite when block applications are fixed, but it does not lead prediction likelihood. Taken together, the two comparisons show that stage granularity changes the trained model: the leading value of L changes between the two configurations and between suite score and prediction loss.



(a) Two repeats (b) Four repeats (c) Four-repeat NLL

Figure 2: Controlled stage-granularity comparisons at width 608. Panel (a) reports B_{suite} for $L \in \{2, 4, 8\}$ at $(M_{\text{mid}}, D_{\text{unique}}, D_{\text{exec}}, R) = (32, 36, 68, 2)$. Panels (b,c) report B_{suite} and NLL for $L \in \{1, 4, 64\}$ at $(64, 68, 260, 4)$. Parameters and processed tokens are matched within each setting; checkpoint parameter counts differ by at most 0.05% within a setting. Panels (a,b) share a 30–80 B_{suite} axis; panel (c) uses a separate NLL axis, on which lower is better. Candidates are ordered by L within each panel.

4.3 Compact-model comparison at matched storage and tokens

The fixed-depth comparisons establish that trained stage groupings differ in quality. We now compare the selected $L = 4$ checkpoint with compact dense and recurrent alternatives at approximately matched parameter storage and processed tokens, alongside the $L = 1$ and $L = 64$ controls. Table 1 also reports a separate width-1024 cohort analyzed in Section 4.4. We then examine the width-608 score by diagnostic domain and on established tasks. Public-model context appears in Appendices G and H.

Four-block LayerLoop leads the tested width-608 models on the suite. At width 608, LayerLoop obtains $B_{\text{suite}} = 71.23$ in Table 1, ahead of the fixed-depth $L = 1$ and $L = 64$ controls (55.39 and 57.51), Dense (46.73), and all four recurrent baselines. Parcae is the strongest of those recurrent baselines at 51.15. This comparison identifies a substantial suite advantage for intermediate four-block stages within the tested storage and data budget. Prediction loss ranks the models differently: $L = 1$ has the numerically lowest NLL estimate (2.800), while Parcae has the lowest NLL estimate among the external recurrent baselines (2.823).

The suite advantage spans several diagnostic domains. Figure 3 decomposes the five baseline rows and the $L = 4$ LayerLoop row from the width-608 cohort of Table 1 into six domains. LayerLoop leads all four recurrent baselines in calibration, context/retrieval, reasoning/consistency, and generation/structure/robustness. It also exceeds the storage- and processed-token-matched dense model in every domain. These gains across domains show that the aggregate advantage is not confined to one kind of diagnostic. Appendix Table 10 reports named raw accuracies.

Established tasks reveal a different ordering. We also evaluate the same checkpoints on seven public benchmark families and a separate frozen controlled-task set. The leading checkpoint varies by task, so these evaluations do not reproduce a single suite-based ranking. Appendix Tables 11 and 12 report the task scores and paired comparisons.

4.4 Suite gains across stored depths and width

Table 1 establishes a suite advantage for four-block LayerLoop over Dense at width 608. We test its extent across five stored depths at that width and in a separate width-1024 cohort, matching stored parameters and processed tokens within each pair.

The suite advantage spans all tested stored depths. With $(L, R) = (4, 4)$ fixed, the width-608 pairs span $M_{\text{mid}} \in \{16, 24, 32, 48, 64\}$ and approximately 150M–434M parameters.

Table 1: Final checkpoints in two storage- and processed-token cohorts. At width 608, $L = 1$ and $L = 64$ are fixed-depth stage-granularity controls, while the other non-LayerLoop rows are shared-recipe baselines. At width 1024, Dense, $L = 2$, and $L = 4$ share a separate budget. Executed depth is not matched across methods; these are not compute-matched comparisons. NLL is lower-is-better, and B_{suite} averages 24 reference-relative percentiles. Bold marks the best value within each cohort. Appendix Table 8 gives exact configurations.

Model	D_{exec}	NLL ↓	B_{suite} ↑
Width 608: approximately 434M parameters, 4.34B tokens			
Dense	68	2.826	46.73
Huginn	260†	3.204	33.19
Mixture-of-Recursions	32	2.945	45.73
Ouro	96	2.866	48.81
Parcae	40	2.823	51.15
LayerLoop ($L = 1$)	260	**2.800**	55.39
LayerLoop ($L = 64$)	260	2.831	57.51
LayerLoop ($L = 4$)	260	2.841	**71.23**
Width 1024: approximately 456M parameters, 4.56B tokens			
Dense	24	2.876	52.08
LayerLoop ($L = 2$)	44	2.811	48.74
LayerLoop ($L = 4$)	44	**2.774**	**62.86**

†Huginn's executed depth is the configured training mean; Mixture-of-Recursions is counted before token routing.

LayerLoop leads B_{suite} in all five pairs by 4.6–24.5 points (Figure 4). Its measured NLL is lower in the three smaller pairs and higher in the two larger pairs. Each LayerLoop model repeats its middle blocks four times, so executed depth differs from Dense. Appendix I (Figure 10) shows the wider width-608 configuration inventory.

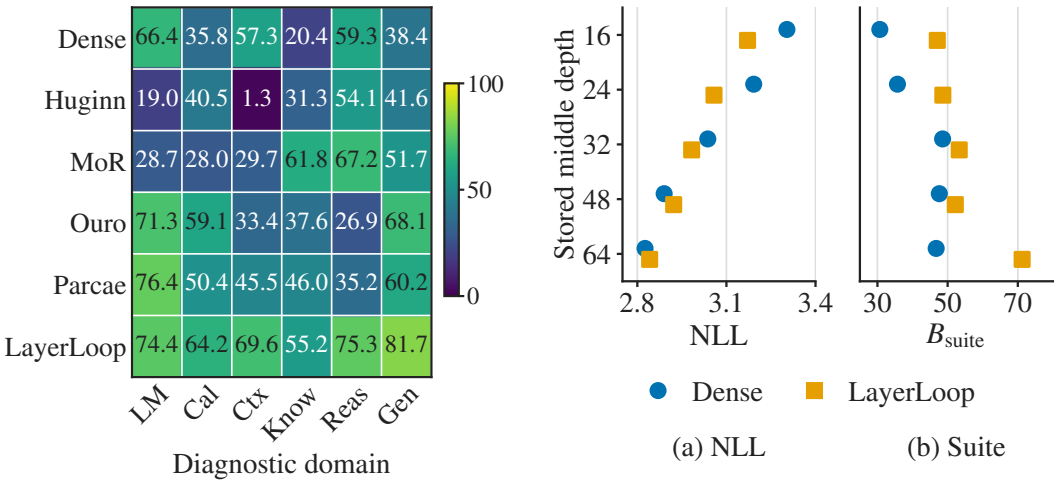


Figure 3: Six-domain profiles for the five width-608 baselines and width-608 $L = 4$ LayerLoop from Table 1. Cells are mean reference-relative component percentiles on a 0–100 scale (higher is better). LM, Cal, Ctx, Know, Reas, and Gen abbreviate language modeling/tail reliability, calibration, context/retrieval, knowledge/adaptation, reasoning/consistency, and generation/structure/robustness. The aggregate weights 24 components equally, not six domains.

Figure 4: Matched width-608 final checkpoints across five stored middle depths. Panels share depth rows and order: (a) held-out NLL (lower is better), and (b) diagnostic suite score (higher is better), with separate horizontal scales. LayerLoop executes four times as many middle-block applications.

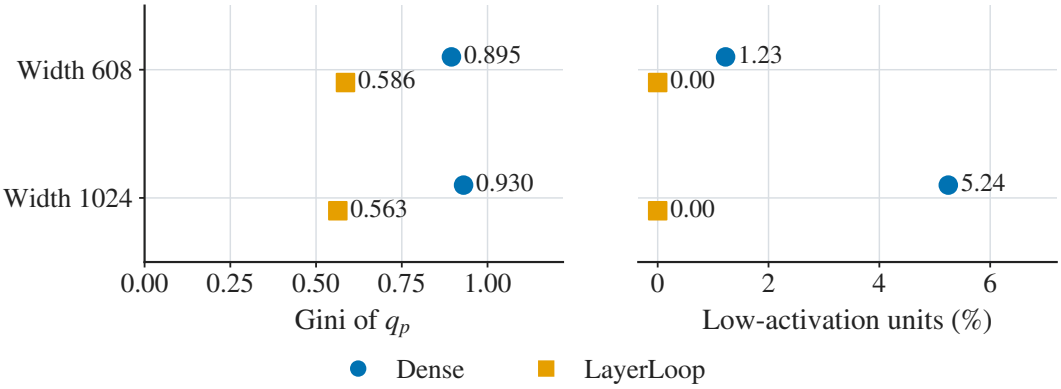
The suite advantage also appears at width 1024. The width-1024 cohort in Table 1 has approximately 456M parameters and 4.56B processed tokens. Four-block LayerLoop has higher suite score and lower NLL than Dense and the two-block-stage model, while executing 44 rather than Dense's

24 blocks. The $L = 2$ result also shows that the suite gain is not shared by every recurrent schedule at this width. Appendix D (Figure 8) provides the matched-depth plot; Appendix J (Figure 11) provides domain profiles and named raw accuracies.

4.5 LayerLoop shows broad parameter participation

To characterize how the trained model uses its compact parameter store, we compare two storage- and token-matched Dense–LayerLoop pairs using first-order loss sensitivity over stored parameters and the fraction of rarely active multilayer-perceptron (MLP) units. Lower sensitivity concentration and fewer rarely active units measure broader participation across stored parameters; Appendix L defines both measurements and their held-out sample.

LayerLoop has lower sensitivity concentration and fewer low-activation units in both widths (Figure 5). The Gini coefficient over stored-parameter sensitivities falls from 0.895 to 0.586 at width 608 and from 0.930 to 0.563 at width 1024. The fraction of low-activation MLP units falls from 1.23% and 5.24% in the respective Dense models to 0.00% in both LayerLoop models. Appendix Figure 12 compares participation across additional stage granularities.



(a) Sensitivity concentration (b) Low-activation units

Figure 5: Parameter participation in two matched Dense–LayerLoop pairs. Left: Gini concentration of first-order loss sensitivity over stored parameters (lower means broader participation). Right: fraction of stored MLP units classified as low-activation using the threshold in Appendix L. The panels share model-width rows and order, but use different horizontal metrics and scales; LayerLoop executes more block applications.

4.6 Trained stages produce repeatable intervention responses

We next ask whether ordered stages produce distinct task responses. We bypass each stage's repeated blocks while retaining its boundary residual and normalization, then compare task-and-loss responses across three independently sampled evaluation sets. The stages received no predefined roles; Appendix M defines the 22-coordinate response profiles and comparison protocol.

In the fully trained checkpoint, same-stage profiles have median cross-sample cosine similarity 0.661 versus -0.002 for different stages; matching profiles identifies the correct stage in 59 of 96 queries (61.5%, versus 6.25% chance). The within-checkpoint pattern also appears at 300 million processed tokens (Appendix Figures 13 and 15). These responses distinguish stages across evaluation samples without assigning exclusive semantic roles.

Replacing one complete trained stage with another among four tested stages increases NLL in all 12 directed swaps (Appendix M); those stages are not freely interchangeable within the fully trained checkpoint. Appendix M also reports robustness and cross-checkpoint alignment tests.

5 Conclusion

LayerLoop makes the size of each repeated stage an explicit architectural choice. In the trained four-repeat comparison, its intermediate four-block stages lead the diagnostic suite at fixed stored and executed depths, while blockwise recurrence has a numerically lower prediction-loss estimate. The selected model also shows broad participation across stored parameters and differentiated responses among stages assigned no roles in advance. Together, these results show that stage granularity provides information about a trained recurrent-depth model beyond its stored and executed depth counts.

References

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. *International Conference on Learning Representations*, 2025a. URL https://arxiv.org/abs/2410.20672.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025b. URL https://arxiv.org/abs/2507.10524.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. *Proceedings of ICML*, 2023. URL https://arxiv.org/abs/2304.01373.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018. URL https://arxiv.org/abs/1807.03819.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020. URL https://arxiv.org/abs/2101.00027.

Jonas Geiping et al. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025. URL https://arxiv.org/abs/2502.05171.

Gemma Team. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. URL https://arxiv.org/abs/2403.08295.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. OLMo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*, 2024. URL https://arxiv.org/abs/2402.00838.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7038–7051, 2021. doi: 10.18653/v1/2021.emnlp-main.564. URL https://aclanthology.org/2021.emnlp-main.564/.

Niall Hurley and Scott Rickard. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741, 2009. doi: 10.1109/TIT.2009.2027527. URL https://arxiv.org/abs/0811.4706.

Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. *arXiv preprint arXiv:2602.11451*, 2026.

Hayden Prairie, Zachary Novack, Taylor Berg-Kirkpatrick, and Daniel Y. Fu. Parcae: Scaling laws for stable looped language models. *arXiv preprint arXiv:2604.12946*, 2026. URL https://arxiv.org/abs/2604.12946.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Kristian Schwethelm, Daniel Rueckert, and Georgios Kaissis. How much is one recurrence worth? iso-depth scaling laws for looped language models. *arXiv preprint arXiv:2604.21106*, 2026. URL https://arxiv.org/abs/2604.21106.

Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. doi: 10.2307/1412159. URL https://www.jstor.org/stable/1412159.

Sho Takase and Shun Kiyono. Lessons on parameter sharing across layers in transformers. *arXiv preprint arXiv:2104.06022*, 2021. URL https://arxiv.org/abs/2104.06022.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL https://arxiv.org/abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/abs/1706.03762.

Abbas Zeitoun, Lucas Torroba-Hennigen, and Yoon Kim. Hyperloop transformers. *arXiv preprint arXiv:2604.21254*, 2026.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025. URL https://arxiv.org/abs/2510.25741.

A Training and Evaluation Details

This appendix provides the data preparation, input construction, training settings, and sample counts needed to interpret Section 4.1. We distinguish recorded configuration values from historical run details that are no longer available.

Dataset and tokenizer. We use a prepared memory-mapped corpus built from `EleutherAI/the_pile_deduplicated`. Preprocessing uses the GPT-NeoX tokenizer from `EleutherAI/pythia-410m-deduped` at revision `main`. The tokenizer contains 50,277 entries and the model vocabulary is padded to 50,304. Token IDs are stored as `uint16`. The prepared training and validation memmaps contain 2,200,000,000 and 8,388,608 tokens, respectively. The preparation script streams the dataset training split once, writes the first 8,388,608 tokens to `val.bin`, and then writes `train.bin` from subsequent documents. The resulting memmaps are sequential and document-disjoint under this script, so we call `val.bin` an internal stream holdout rather than an official validation split. A sampled overlap audit hashed 34,254,712 length-64 training windows at stride 64 and found zero matches among 512 validation windows. This sampled audit does not prove corpus-level independence, so external-corpus evaluation remains a useful prospective check.

Models use sequences of 2,048 tokens. Several reported processed-token counts exceed the prepared training memmap because the loader wraps and revisits tokens; exact restart and shuffle behavior is not uniformly retained.

Held-out prediction loss. The reported negative log-likelihood (NLL) is the unweighted mean of length-specific per-token cross-entropy at sequence lengths 512, 1,024, and 2,048. Each length-specific value pools token losses over sequences sampled from the internal holdout. In the compact-model comparison, Dense uses 64 sequences per length and each other model uses 128, drawn from the same holdout but not paired by sequence. The small differences between model-specific NLL estimates are therefore reported descriptively, not as paired-sample effects.

Public-reference evaluation. Public checkpoints use the same frozen evaluation-suite implementation, Pile validation memmap, and 58-row score reference pool. Pythia shares the local tokenizer directly. OLMo has a 50,280-entry tokenizer; 50,276 of the 50,277 local vocabulary IDs align with it (99.998% of local IDs). OLMo text cases are therefore constructed with the official Pythia Hugging Face tokenizer, while the official OLMo tokenizer is retained for compatibility checks. The legacy OLMo checkpoint code runs in an isolated Transformers 4.39.3 compatibility environment; model weights and task definitions are unchanged.

Shared optimization recipe. Archived configurations for our locally trained models use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1, and gradient-norm clipping at 1.0. The peak learning rate is 3×10^{-4}. It increases linearly during the first 2% of the target token budget and then follows cosine decay to 3×10^{-5}. Training uses bfloat16 arithmetic and no synthetic-data mixture. Available configs schedule validation every 52,428,800 tokens and checkpointing every 104,857,600 tokens. Because complete run-specific launch metadata are not retained for every historical run, these values describe the shared recipe supported by available records rather than a claim that every run-specific field has been recovered.

Stored-parameter accounting. Let P_{blk} be the parameters in one Transformer block and P_{base} include embeddings, final normalization, and other parameters outside the stored blocks and stage operations. Each of the K LayerLoop stages stores R width-d repeat-conditioning vectors, R learned residual-scale scalars, and d stage-boundary-normalization gains. Thus $P_{\text{stage}} = K(Rd + R + d)$, and the stored count and a hypothetical unshared-execution count are

$$P_{\text{stored}} = P_{\text{base}} + D_{\text{unique}} P_{\text{blk}} + P_{\text{stage}}, \qquad P_{\text{unshared-exec}} = P_{\text{base}} + D_{\text{exec}} P_{\text{blk}} + P_{\text{stage}}. \quad (5)$$

The experiments report checkpoint-measured parameter counts; this formula separates stored capacity from repeated execution without treating the unshared count as a trained model.

Table 2: Recorded runtime and update budgets for eight local training runs. LayerLoop tuples give (K, L, R). Tokens/update equals sequence length times world size times microbatch times gradient accumulation. Wall time, GPU-hours, and average throughput are archival cost records rather than a cross-row systems comparison because hardware, world size, and memory settings differ.

Run	Tokens	$D_{\text{unique}}/D_{\text{exec}}$	GPUs	Tokens/update	Wall-h	GPU-h	tok/s
Dense-24, width-1024	4.563B	24/24	4	131,072	14.65	58.60	86.5k
LayerLoop-1024 $(5, 4, 2)$	4.563B	24/44	6	221,184	17.36	104.16	73.0k
LayerLoop-1024 $(10, 2, 2)$	4.564B	24/44	4	131,072	21.37	85.48	59.3k
LayerLoop-768 $(10, 4, 2)$	4.556B	44/84	6	221,184	21.76	130.56	58.2k
Dense-68, width-608	4.341B	68/68	3	98,304	36.48	109.44	33.1k
LayerLoop-608 $(16, 4, 4)$	4.342B	68/260	6	196,608	46.48	278.88	25.9k
LayerLoop-608 $(64, 1, 4)$	4.343B	68/260	6	196,608	47.07	282.42	25.6k
LayerLoop-608 $(1, 64, 4)$	4.341B	68/260	6	196,608	56.18	337.05	21.5k

For every row in Table 2, the prepared corpus identity and processed-token total are confirmed. Exact example order, restart boundary, and cycling policy are unknown in the retained historical metadata. The comparisons therefore claim corpus and processed-token matching, not identical example order.

The complete machine-readable archive records microbatch, gradient accumulation, peak reserved memory when available, checkpoint coverage, and partial continuation segments for models whose from-zero wall time could not be recovered. Across the 24 trained Pythia/Pile architectures, eight have trustworthy end-to-end runtime records, five additional rows have a verified continuation segment, and the remaining rows retain parameter and token budgets only. Missing wall times are left missing rather than reconstructed from heterogeneous hosts or instantaneous throughput.

Checkpoint endpoints and replication policy. Both `best.pt` and `final.pt` are archived for each trained Pythia-class row used in the paper. Headline comparisons use final endpoints. Graph-edit diagnostics contain edits of both best and final sources and treat edits of the same source as correlated. The exact validation criterion used to label older checkpoints "best" and the numerical training seeds are not uniformly retained, so no primary claim is based on treating best/final as independent runs. Compute constraints allowed one training run per architecture–configuration pair; we do not compute across-seed intervals or significance tests over training randomness.

Evaluation suite and available task sizes. We evaluate checkpoints with a frozen implementation of the 24-component suite. The archive contains 58 unmodified reference checkpoints and 76 graph-edit targets, as defined in Appendix E. Representative evaluation summaries recover the task-level counts in Table 3. Raw-LM and context-conditioning measures are token- or bucket-aggregated and are not represented as question counts. Any example-level interval would be conditional on the fixed trained checkpoint and would not measure variation across retraining.

Table 3: Evaluation sizes for suite sections. Pair- or prompt-derived observations are shown at their natural clustered unit where available.

Section	Size	Section	Size
Text structure	192 examples	Forced choice	160 examples
Few-shot pattern	60 examples	Prompt robustness	480 derived examples
Copy induction	64 examples	Counterfactual	48 examples / 24 pairs
Order sensitivity	72 examples	Generation diagnostics	64 samples
Sampled generation	72 samples	Reading comprehension	72 examples
Needle in context	108 examples	Truthfulness	72 examples
Analogical mapping	72 examples	Definition recall	72 examples
Code reasoning	72 examples	Knowledge	96 examples
Controlled synthetic	16×3 settings	Raw LM/context conditioning	token/bucket aggregates

Metadata caveats. The checkpoint archive is complete for the reported Pythia-class rows, but complete launch metadata and full logs are not uniformly present. Missing run-specific fields include some attention/MLP details, numerical seeds, hardware model, and data-loader cycling behavior. The evaluation record is separate from these missing training details: Appendix B defines the percentile tie convention, and all primary score rows have complete component coverage. The archived component values, direction map, and scoring script specify how each reported aggregate is calculated.

Inference timing protocol. We measure the width-608 and width-1024 Dense–LayerLoop pairs on one NVIDIA L40 using bfloat16 arithmetic and batch size one. Prefill is median prompt-processing latency over 100 measurements at 1,024 tokens; decode is median per-token latency over 50 timed trials of 64-token greedy continuations from 512-token prompts. Generation recomputes the complete context at every step, without a key–value cache; warmups are excluded. At width 608, Dense/LayerLoop prefill medians are 39.83/153.77 ms and decode medians are 38.28/153.58 ms per token. At width 1024, the corresponding medians are 13.43/24.14 ms and 13.56/24.62 ms per token. Dense/LayerLoop p95 prefill and decode latencies are 40.25/154.79 and 38.88/154.75 ms at width 608, and 13.84/24.49 and 13.79/25.08 ms at width 1024. These timings characterize this uncached implementation and do not estimate latency under cached or optimized generation.

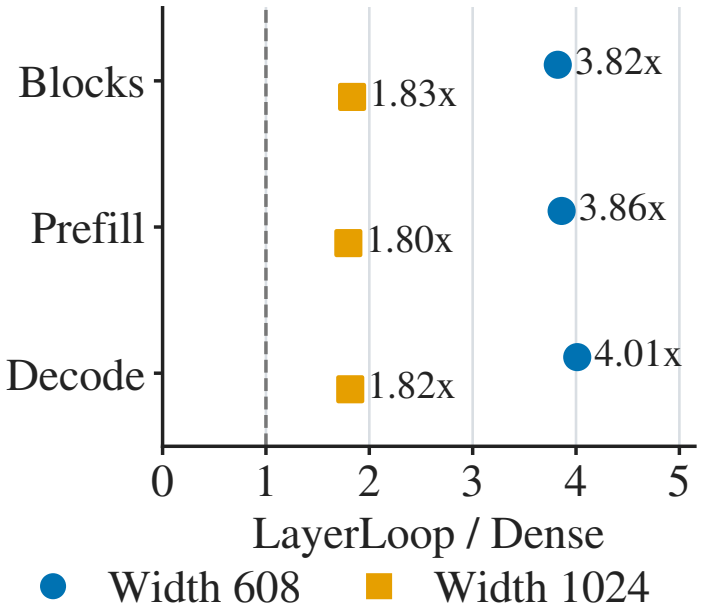


Figure 6: LayerLoop-to-Dense ratios for block applications, prefill, and decode at widths 608 and 1024. Block ratios use executed depth; latency ratios use measured medians on one L40 with batch size one and full-context recomputation without a key–value cache.

B Broad-Coverage Suite Components

This appendix makes the suite in Section 4.1 auditable. We list the 24 components and their aggregation rules, test sensitivity to the reference pool, and define the six reporting domains. Appendix J reports the resulting domain profiles and named raw tasks with their model comparisons. Each component receives equal weight in B_{suite}; reporting domains contain different numbers of components and therefore contribute different total weights. The frozen pool contains 12 internally trained Dense checkpoints, 34 internally trained LayerLoop checkpoints, and 12 public Pythia-410M trajectory checkpoints. Every pool member has complete component coverage.

Evaluator field names are preserved for all component codes. In these codes, "acc" denotes the fraction of correct examples, pointwise mutual information (PMI) normalization subtracts an answer-only log score from each prompted choice log score and divides by the candidate's prompted-token count. This follows the likelihood-ratio principle used for candidate scoring (Holtzman et al., 2021); the answer-only context here is the literal prompt `Answer:`. Expected calibration error (ECE) measures the confidence–accuracy gap (Guo et al., 2017): we split predictions into ten confidence bins and average their absolute binwise gaps, weighted by the number of predictions in each bin. Negative log-likelihood (NLL) is averaged over the stated token set. Table 4 gives the remaining aggregation details and the complete mapping into the six reporting domains.

Table 4: Frozen evaluation-suite components used by B_{suite}. Arrows give the raw-metric direction before percentile conversion.

Component	Plain-language statistic	Reporting domain	Dir.
analogy_symbol_acc	Accuracy on symbolic analogies	Reasoning/consistency	↑
choice_pmi_ece	Ten-bin ECE of PMI-normalized forced-choice predictions	Calibration	↓
code_ece	Ten-bin ECE on code-reasoning choices	Calibration	↓
controlled_long_acc	Mean token accuracy in the long member of controlled context pairs	Context/retrieval	↑
long_context_gain_loss	Local-context suffix NLL minus full-context suffix NLL at the longest tested length	Context/retrieval	↑
copy_induction_ece	Ten-bin ECE on copy-induction choices	Calibration	↓
counterfactual_acc	Accuracy on counterfactual questions	Reasoning/consistency	↑
definition_recall_acc	Accuracy on definition-recall questions	Knowledge/adaptation	↑
few_shot_pmi_acc	PMI-normalized accuracy on few-shot pattern questions	Knowledge/adaptation	↑
max_repeated_token_run	Mean longest run of one identical token in each greedy generation	Generation/structure/robustness	↓
knowledge_pmi_acc	PMI-normalized factual-knowledge accuracy	Knowledge/adaptation	↑
order_ece	Ten-bin ECE on order-sensitivity choices	Calibration	↓
robust_clean_worst_delta	Clean accuracy minus the lowest accuracy over prompt perturbations	Generation/structure/robustness	↓
raw_bits_per_byte	Sum of held-out token negative log probabilities divided by the corresponding UTF-8 byte count and $\ln 2$	Language modeling/tail	↓
raw_p95_token_loss	95th percentile of held-out token NLL	Language modeling/tail	↓
raw_p99_token_loss	99th percentile of held-out token NLL	Language modeling/tail	↓
sampled_temp07_rep	Mean $1 -$ unique tokens$/$generated tokens at temperature 0.7 and top-$p = 0.9$	Generation/structure/robustness	↓
text_answer_loss	Mean per-token NLL on forced text-structure answer spans	Generation/structure/robustness	↓
text_exact	Exact-match accuracy on text-structure answers	Generation/structure/robustness	↑
reading_coreference_acc	Accuracy on the coreference subset of reading questions	Reasoning/consistency	↑
needle_long_acc	Needle-retrieval accuracy at 2,048 tokens ($n = 36$)	Context/retrieval	↑
counterfactual_pair_consistency	Fraction of original/counterfactual pairs answered consistently	Reasoning/consistency	↑
needle_acc	Needle-retrieval accuracy across all tested lengths (108 examples total)	Context/retrieval	↑
truthfulness_acc	Accuracy on truthfulness questions	Reasoning/consistency	↑

Frozen percentile definition. To combine the differently scaled components above, we convert each raw value to a percentile against a fixed reference pool. Let $\mathcal{R}$ contain the 58 reference checkpoints and let $\mathcal{C}$ contain the 24 components in Table 4. For component c, orient its raw value so that a larger value is better. Given target checkpoint m, let $n_<(m, c)$ and $n_=(m, c)$ count reference values strictly below and equal to the oriented target value. The counts refer to checkpoints in $\mathcal{R}$, including the target when it is a reference member. The indicator $\mathbf{1}\{\cdot\}$ equals one when its condition is true and zero otherwise. We use the average-rank percentile extension:

$$\pi_{m,c} = \frac{100}{|\mathcal{R}|}\left(n_<(m,c) + \frac{n_=(m,c)}{2} + \frac{\mathbf{1}\{n_=(m,c) > 0\}}{2}\right), \qquad B_{\text{suite}}(m) = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \pi_{m,c}. \tag{6}$$

For reference members this is 100 times their average rank divided by 58; off-reference values use their position relative to the same anchors. This is a stepwise rank, so distinct values between adjacent anchors can receive the same percentile. Tied values receive identical scores. We apply this rule to new recurrent baselines and public reference comparisons without inserting them into $\mathcal{R}$. All compact-model rows cover all 24 components. The score is a mean reference-relative percentile, not a public benchmark accuracy or a ratio-scale measure of model ability; differences in score points are the natural comparisons.

Reference-pool sensitivity. Because B_{suite} averages reference-relative percentiles, changing the anchor pool can change its absolute value. We recompute all 24 component percentiles against six pools using the average-rank extension defined above, without rerunning inference (Table 5). The five alternate anchor pools retain positive LayerLoop-minus-Dense gaps for all five width-608 pairs. The width-1024 gap is also positive, ranging from 9.20 to 11.28 score points across those five pools. Rank correlation with the full pool remains at least 0.930 after rounding; the Pythia-only pool is a deliberately small stress test rather than an alternative score definition.

Table 5: Sensitivity of B_{suite} to the percentile anchor pool. Correlation is Spearman rank correlation, the correlation between the two checkpoint rankings (Spearman, 1904), over all 58 reference checkpoints against their full-pool scores. The delta range is LayerLoop minus dense across the five width-608 matched pairs.

Anchor pool	Rows	Rank corr.	Width-608 wins	Delta range
Full reference pool	58	1.000	5/5	+4.56 to +24.50
Internal checkpoints	46	0.996	5/5	+5.30 to +26.59
Pythia-410M ladder	12	0.930	5/5	+1.74 to +16.32
Final/single + Pythia	36	0.995	5/5	+4.17 to +23.50
Dense + Pythia	24	0.988	5/5	+2.52 to +21.18
LayerLoop + Pythia	46	0.997	5/5	+4.76 to +24.00

Reporting domains. For interpretability, we group all 24 components into six capability domains and average the same direction-aligned percentile ranks within each domain. For reporting domain g, Table 4 gives $\mathcal{C}_g \subseteq \mathcal{C}$, and its mean is $|\mathcal{C}_g|^{-1} \sum_{c \in \mathcal{C}_g} \pi_{m,c}$ for checkpoint m. Appendix J reports the trained-checkpoint domain profiles and named raw accuracies in Figure 11.

C Pythia-410M Training Checkpoints

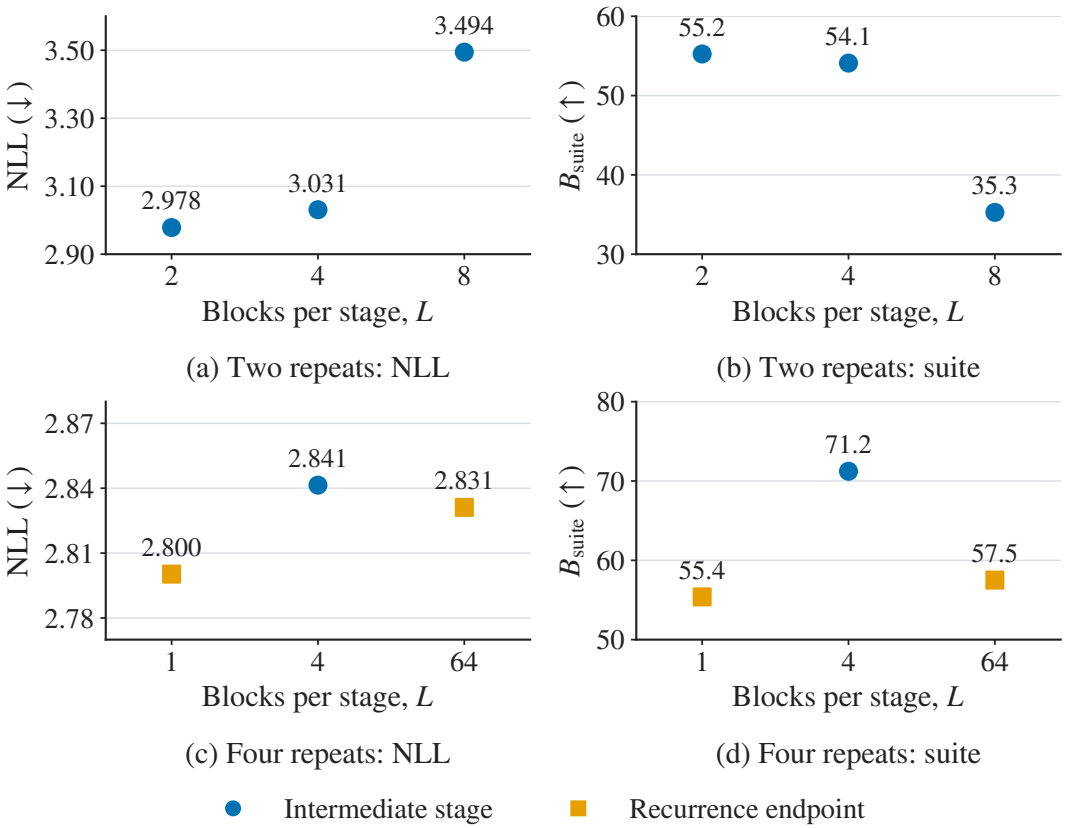
Table 6 lists the 12 public Pythia-410M checkpoints included in the frozen reference pool of Section 4.1. Later checkpoints have lower NLL, but their diagnostic-suite score fluctuates. This distinction explains why the threshold analysis in Appendix G uses the first observed crossing rather than assuming a monotone score trajectory. Checkpoint names identify optimizer steps.

Table 6: NLL decreases while B_{suite} fluctuates along the Pythia-410M training trajectory. These checkpoints supply public reference ranks rather than matched architectural controls.

Checkpoint	Tokens	$B_{\text{suite}} \uparrow$	NLL $\downarrow$
step1000	2.10B	20.58	3.974
step2000	4.19B	30.24	3.294
step8000	16.78B	57.79	2.736
step16000	33.55B	63.22	2.618
step24000	50.33B	61.67	2.569
step32000	67.11B	69.58	2.539
step40000	83.89B	66.49	2.517
step48000	100.66B	66.13	2.502
step56000	117.44B	72.16	2.489
step64000	134.22B	67.13	2.476
step88000	184.55B	69.61	2.442
step112000	234.88B	66.92	2.430

D Detailed Granularity Comparisons

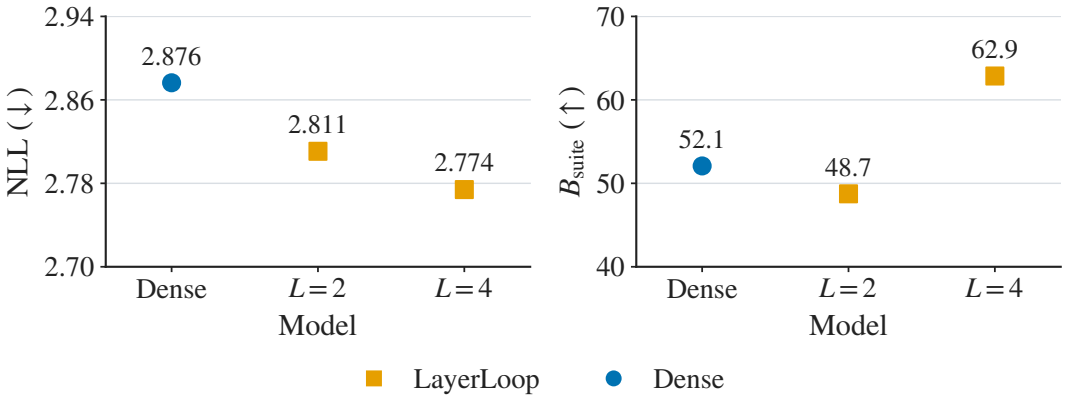
Using Section 3's notation for middle blocks (M_{mid}), stored depth (D_{unique}), executed depth (D_{exec}), and repeat count (R), these figures give both NLL and suite score for the granularity comparisons in Sections 4.2 and 4.4. Two-block stages lead both measures in the width-608 two-repeat setting (Figure 7a,b). In the four-repeat setting, blockwise recurrence has the lowest NLL while four-block stages have the highest suite score (Figure 7c,d). At width 1024, four-block stages have both the lowest NLL and the highest suite score (Figure 8).



(a) Two repeats: NLL (b) Two repeats: suite

(c) Four repeats: NLL (d) Four repeats: suite

Figure 7: Width-608 stage-granularity details. The top row compares $L = 2, 4, 8$ at $(M_{\text{mid}}, D_{\text{unique}}, D_{\text{exec}}, R) = (32, 36, 68, 2)$; the bottom compares $L = 1, 4, 64$ at $(64, 68, 260, 4)$. Within each row, NLL (left) and B_{suite} (right) share the candidate order; parameter counts are closely matched and processed-token budgets are matched. The rows use different depth and repeat settings and different numeric ranges. Lower NLL and higher B_{suite} are better. Orange squares mark recurrence endpoints in the bottom row.



(a) Held-out NLL (b) Diagnostic suite

Figure 8: Four-block stages lead both measures in the width-1024 comparison. All models have approximately 456.3M parameters, process 4.563–4.564B tokens, and store 24 blocks with distinct parameters; the two LayerLoop rows execute 44 block applications. Dense executes 24 blocks. All are final checkpoints; $L = 2$ denotes $(K, L, R) = (10, 2, 2)$ and $L = 4$ denotes $(5, 4, 2)$. The panels share the model order, use different metric scales, and display the plotted ranges.

E Fixed-Weight Graph-Edit Stress Tests

Section 4.2 compares models trained under different stage schedules. This appendix instead asks what happens when an already trained model's execution schedule is changed without retraining. We rewire the computation while keeping stored weights fixed. The edits below change repeated execution or remove repeats. All 76 variants have higher NLL than their source checkpoint, and 72 have lower B_{suite}, indicating sensitivity to the training schedule. Edits derived from the same source checkpoint are not independent observations. We first define the edits and their source groups, then define the degradation measures and report the complete grouped results in Table 7.

For dense sources, each edit applies the first and last two blocks once, groups the middle blocks into K units of L blocks, and applies each unit $R = 2$ times without retraining. These edits reuse the original blocks at unit residual scale and add no repeat-conditioning vectors, stage-boundary residuals, normalization gains, or other parameters. "Each-layer" uses $L = 1$ and $K = M_{\text{mid}}$; "$L = 4$" uses $K = M_{\text{mid}}/4$; and "middle" uses $K = 1$ and $L = M_{\text{mid}}$. The four "matched" edits reproduce the (K, L, R) schedules of the paired trained LayerLoop controls: $(4, 8, 2)$ at width 608 and $(10, 2, 2)$ at width 1024, each at its best and final checkpoint. A repeat-to-dense-once edit applies every trained stage block stack once and bypasses stage repeats, repeat-conditioning vectors, repeat-specific residual scales, and the stage-boundary residual and stage-boundary normalization. Thus the table's edit count includes best and final endpoints and does not count independent trained models.

The three general dense-source groups each include best and final checkpoints from six models: five width-608 models with $M_{\text{mid}} \in \{16, 24, 32, 48, 64\}$ and one width-1024 model with $M_{\text{mid}} = 20$. The matched group uses two of these sources at both checkpoints. The repeat-to-dense group includes 18 recurrent configurations at two checkpoints each.

Formally, let e be a fixed-weight graph edit applied to source checkpoint m, let ℓ_m denote its held-out NLL, and let $b_m = B_{\text{suite}}(m)$ denote its behavioral suite score. We define

$$\Delta_\ell(e; m) = \ell_{e(m)} - \ell_m, \qquad \Delta_b(e; m) = b_{e(m)} - b_m. \tag{7}$$

Thus positive Δ_ℓ and negative Δ_b indicate degradation. Both source and edited models are scored against the fixed 58-checkpoint pool using Appendix B's average-rank rule. Within each table group, means and medians summarize one difference per edited checkpoint, with equal weight per edit; the worse columns give the fraction with the degradation sign. The table contains 76 edits, not 76 independently trained models.

Table 7: Fixed-weight graph-edit stress tests. Positive Δ_ℓ and negative Δ_b indicate degradation. Rows summarize correlated edits of the source checkpoints. The fixed 58-row reference pool contains the unmodified checkpoint rows used as percentile anchors.

Edit group	Edits	Mean Δ_ℓ	Med. Δ_ℓ	NLL worse	Mean Δ_b	Med. Δ_b	b worse
Dense → each-layer repeat	12	+3.620	+1.138	100%	-23.806	-22.162	100%
Dense → $L = 4$ stage repeat	12	+3.007	+0.750	100%	-23.958	-23.761	100%
Dense → matched stage repeat	4	+4.494	+4.350	100%	-24.318	-28.197	100%
Dense → whole-middle repeat	12	+1.475	+0.247	100%	-7.663	-2.065	66.7%
Repeated → dense-once	36	+5.178	+4.940	100%	-33.051	-32.813	100%

F Recurrent Baselines and Additional Evaluations

This appendix documents the recurrent comparisons in Section 4.3. We specify which mechanisms are retained, define the checkpoint selection policy, report the established benchmarks and controlled probes, and document the incomplete LoopFormer run. The suite components and percentile rule are defined in Appendix B.

Shared training protocol. The four completed recurrent baselines use the prepared Pile stream, tokenizer, context length, AdamW recipe, and 196,608-token update batch documented in Appendix A. Each uses training seed 520 and processes 4,341,694,464 tokens, the same realized budget as the main width-608 LayerLoop checkpoint. Whole update batches may slightly exceed a nominal token target. Parameter counts are taken from the evaluated checkpoints. Our implementations use common embeddings, causal attention, and SwiGLU blocks, while retaining each baseline's

published recurrence, routing, injection, or output-combination mechanism illustrated in Figure 9. They are shared-recipe implementations, not reproductions of the original published training runs.

Table 8 gives the exact sizes and depths of all main-table models, including the two stage-granularity controls.

Table 8: Configuration details for the final checkpoints in Table 1. Parameter counts come from evaluated checkpoints. All rows use the same prepared corpus, tokenizer, and evaluation suite. D_{unique} counts distinct stored blocks; D_{exec} counts block applications, except Huginn's configured training mean. Mixture-of-Recursions is counted before token routing.

Model	Width	Params (M)	Tokens (B)	D_{unique}	D_{exec}
Dense	608	434.11	4.341	68	68
Huginn	1344	417.21	4.342	12	260
Mixture-of-Recursions	1360	426.36	4.342	12	32
Ouro	960	452.54	4.342	24	96
Parcae	1360	430.06	4.342	12	40
LayerLoop ($L = 1$)	608	434.30	4.343	68	260
LayerLoop ($L = 64$)	608	434.11	4.341	68	260
LayerLoop ($L = 4$)	608	434.16	4.342	68	260
Dense	1024	456.31	4.563	24	24
LayerLoop ($L = 2$)	1024	456.34	4.564	24	44
LayerLoop ($L = 4$)	1024	456.33	4.563	24	44

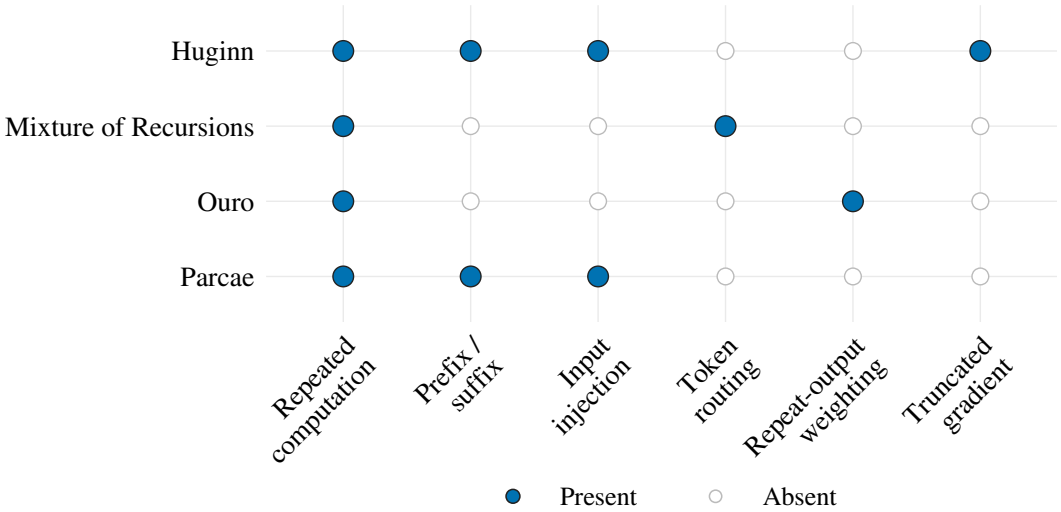


Figure 9: Mechanisms retained in the recurrent baseline implementations. A filled mark means that the baseline implementation retains the corresponding mechanism. Repeated computation denotes reuse of stored transformations; prefix/suffix denotes non-recurrent blocks around the repeated middle; input injection includes additive or learned-decay mixing of a recurrent input; token routing is Mixture-of-Recursions' capacity-constrained selection; repeat-output weighting is Ouro's learned exit mixture; truncated gradient is Huginn's stop-gradient prefix and trained suffix. Stored and executed block counts are in Appendix Table 8.

Endpoint policy. Table 1 uses final checkpoints for every completed method. Table 9 additionally reports the available validation-selected Ouro and Parcae checkpoints using the same evaluator. Validation-selected NLL in that table is measured by the final diagnostic evaluator; it need not equal the online validation estimate used for selection. There is one trained model per baseline configuration, so these checkpoint alternatives do not constitute independent training replicates.

Table 10 provides the raw task accuracies behind selected diagnostic domains. The different winners across tasks complement the aggregate comparison: LayerLoop ties the highest knowledge and truthfulness accuracies here, while Dense leads reading.

Table 9: Available validation-selected baseline checkpoints. These supplement the final-checkpoint compact-model comparison; they are not selected by suite score.

Model	Tokens (B)	NLL ↓	B_{suite} ↑
Ouro	2.779	2.968	43.61
Parcae	3.775	2.844	54.96

Table 10: Raw diagnostic accuracies (percent) for the compact-model comparison. Choice, Knowledge, Reading, Code, and Truth use 160, 96, 72, 72, and 72 internal examples, respectively; Needle 2048 uses 36 examples at the longest context. These are raw task accuracies, distinct from the calibration and pointwise-mutual-information components in Table 4, and are not scores on standard public benchmark datasets.

Model	Choice	Knowledge	Reading	Code	Needle 2048	Truth
Dense	67.5	83.3	83.3	55.6	83.3	58.3
Huginn	55.0	91.7	41.7	33.3	50.0	50.0
Mixture-of-Recursions	52.5	91.7	66.7	33.3	66.7	50.0
Ouro	60.0	75.0	75.0	44.4	83.3	50.0
Parcae	55.0	83.3	50.0	55.6	83.3	50.0
LayerLoop	60.0	91.7	66.7	44.4	83.3	58.3

Established public benchmarks. The internal probes above identify behaviors behind the diagnostic suite. We next evaluate whether their model ordering carries over to established datasets under a shared zero-shot likelihood-scoring protocol. LayerLoop, its $L = 1$ and $L = 64$ granularity controls, Dense, and the four recurrent baselines receive the same 22,632 examples from BoolQ, COPA, SciQ, Social IQA, CommonsenseQA, RACE, and MMLU; MMLU uses no in-context demonstrations. Raw accuracy is available for 63 common task entries, including 57 MMLU subjects. SciQ also has length-normalized accuracy, which selects an answer after dividing each candidate's log likelihood by its token count, giving 64 task–metric pairs per comparison model. For each pair, we test the checkpoints' example-matched binary correctness with the exact McNemar test and apply Holm correction jointly across all $7 \times 64 = 448$ comparisons involving LayerLoop. Table 11 shows that the ordering does not persist: the leading checkpoint changes across tasks. Holm correction over all 448 paired task–metric comparisons involving LayerLoop, including the 57 individual MMLU subjects, retains only four statistically significant differences: its two BoolQ deficits relative to $L = 64$ and Huginn and its SciQ raw-accuracy and length-normalized-accuracy advantages relative to Huginn. The remaining comparisons do not reach the corrected 0.05 level. The complete paired-test rows, including unadjusted and corrected values, are provided in `supplementary/paired_task_comparisons.csv`.

Table 11: Zero-shot public-benchmark accuracy (percent) for the compact checkpoints and the two granularity controls. SciQ-N is length-normalized SciQ accuracy; SIQA and CSQA denote Social IQA and CommonsenseQA; MMLU-0 is the zero-shot MMLU macro average. All rows use the same 22,632 examples. Bold marks the largest value in a column and does not by itself indicate a statistically reliable difference.

Model	BoolQ	COPA	SciQ	SciQ-N	SIQA	CSQA	RACE	MMLU-0
$L = 1$	57.4	**65.0**	69.7	59.9	37.0	19.5	**28.6**	23.1
LayerLoop ($L = 4$)	54.8	**65.0**	68.6	60.6	37.1	19.6	**28.6**	22.9
$L = 64$	59.4	63.0	67.5	60.1	**37.6**	**19.7**	28.0	22.9
Dense	57.8	64.0	65.1	59.2	36.0	19.5	26.7	**23.2**
Huginn	**61.6**	56.0	60.2	51.6	35.9	**19.7**	25.2	23.1
Mixture-of-Recursions	54.7	59.0	65.9	56.4	35.8	**19.7**	27.6	**23.2**
Ouro	57.5	63.0	**69.9**	**61.0**	36.7	19.5	27.8	**23.2**
Parcae	58.0	63.0	66.4	59.2	36.2	19.5	27.3	23.1

Frozen controlled tasks. To evaluate the same checkpoints on a separate controlled task set without redefining the diagnostic suite, we generate 120 unique four-choice examples in each of six programmatic families before inspecting model outputs: definition recall, event order, numeric analogy, code arithmetic, state tracking, and symbol transformation. Every checkpoint receives the same 720 prompts, options, and answer indices. Table 12 reports raw and prior-adjusted accuracy separately; "raw" means without answer-only prior adjustment, while choices are still ranked by

mean token loss. The prior-adjusted score subtracts each option's score under the context-free prompt "Answer:" from its contextual score, then divides by the candidate's prompted-token count before selecting an answer. Using exact McNemar tests of paired binary correctness and Holm correction over the 12 tests within each LayerLoop–baseline comparison, differences remain only against Huginn: LayerLoop is lower on code arithmetic by 12.50 raw and 20.83 prior-adjusted accuracy points, and higher on raw state tracking by 15.00 points. Several code arithmetic and symbol-transformation scores are at or below the 25% chance level. We therefore treat these results as controlled response profiles rather than estimates of general task competence.

Table 12: Accuracy (percent) on six frozen programmatic task families, each containing 120 four-choice examples. Each cell reports raw / prior-adjusted accuracy. All checkpoints use the same 720 examples; the chance level is 25%.

Model	Definition recall	Event order	Numeric analogy	Code arithmetic	State tracking	Symbol transform
$L = 1$	22.5/24.2	20.8/14.2	9.2/17.5	0.8/12.5	32.5/24.2	2.5/0.0
LayerLoop ($L = 4$)	20.0/23.3	24.2/19.2	12.5/15.0	0.0/8.3	25.0/21.7	8.3/1.7
$L = 64$	22.5/23.3	15.8/10.8	5.8/22.5	0.8/17.5	30.0/29.2	3.3/3.3
Dense	19.2/20.0	25.8/10.8	14.2/19.2	1.7/9.2	32.5/25.0	5.8/5.0
Huginn	25.0/20.8	15.0/9.2	13.3/27.5	12.5/29.2	10.0/19.2	3.3/5.0
Mixture-of-Recursions	22.5/24.2	17.5/13.3	11.7/16.7	2.5/22.5	20.8/22.5	5.0/8.3
Ouro	19.2/19.2	20.0/13.3	10.0/17.5	0.8/19.2	13.3/16.7	5.8/5.0
Parcae	24.2/25.0	13.3/10.8	14.2/21.7	0.8/13.3	30.0/30.0	5.8/5.0

Incomplete baseline. The LoopFormer implementation (Jeddi et al., 2026) retains time/step modulation, full and short trajectories, and hidden-state consistency. Two runs, with 2% and 8% warmup, became numerically unstable at approximately 18M tokens. The latter stopped at 18,087,936 processed tokens when a step loss of 116.462 exceeded the stopping threshold. It has no comparable complete-training endpoint and is therefore not assigned a score in the main table. This outcome describes this implementation under the tested recipes.

G Public Model Context

This appendix provides the separate public-model scale context referenced in Section 4.3 and places the main width-608 LayerLoop model and two wider variants among public models. Table 13 compares checkpoints near ten processed tokens per parameter; Table 14 compares fully trained public checkpoints. Both use the same suite and frozen percentile anchors as Section 4.1, while the models' original data and computation budgets differ. The final subsection then reports the first observed Pythia checkpoint to cross selected suite-score thresholds. For checkpoint m, processed tokens per stored parameter is $\mathrm{Tpp}(m) = T_m/P_m$, where T_m and P_m are its processed-token count and stored-parameter count. We compute the ratio before rounding displayed values. It describes data exposure relative to model size, not equal training compute. Pythia revisions are selected nearest ten processed tokens per parameter; OLMo contributes a reference at 10.308 tokens per parameter (Groeneveld et al., 2024). OLMo is evaluated with the common tokenizer under the compatibility procedure in Appendix A. Fully trained Llama-2 (Touvron et al., 2023), Qwen2.5 (Qwen Team, 2024), and Gemma (Gemma Team, 2024) are evaluated with their native tokenizers on the same decoded source text. Their tokenwise losses and aggregate suite scores are not placed alongside the common-tokenizer rows in Table 14; Appendix H identifies the raw text tasks that support narrower cross-tokenizer comparisons. The tables repeat the main width-608 $(K, L, R) = (16, 4, 4)$ endpoint and add two $L = 4$ LayerLoop references: $(K, L, R) = (10, 4, 2)$ at width 768 and $(5, 4, 2)$ at width 1024. The width-608 model exceeds the five Pythia sizes near ten tokens per parameter on B_{suite} and trails OLMo-7B by 1.61 points at 10.31 tokens per parameter. Among common-tokenizer comparisons at full public training budgets, Pythia-6.9B and OLMo-7B score higher, by 1.58 and 1.04 points. These comparisons describe training trajectories rather than attribute their differences to architecture alone.

Table 13: LayerLoop and public base checkpoints near ten processed tokens per stored parameter. LayerLoop tuples give (K, L, R) and all three rows use final checkpoints. Public training data and computation differ; those rows provide scale context.

Model	Params (M)	Tokens (B)	Tpp	NLL ↓	B_{suite} ↑
LayerLoop-1024 $(5, 4, 2)$	456.3	4.563	10.00	2.774	62.86
LayerLoop-608 $(16, 4, 4)$	434.2	4.342	10.00	2.841	71.23
LayerLoop-768 $(10, 4, 2)$	455.5	4.556	10.00	2.763	67.53
Pythia-410M	405.3	4.194	10.35	3.294	30.24
Pythia-1B	1,011.8	10.486	10.36	2.693	61.14
Pythia-1.4B	1,414.6	14.680	10.38	2.553	64.76
Pythia-2.8B	2,775.2	27.263	9.82	2.366	68.86
Pythia-6.9B	6,857.3	69.206	10.09	2.168	64.73
OLMo-7B	6,888.1	71.000	10.31	2.602	72.84

Table 14: LayerLoop and fully trained public base models evaluated with the common GPT-NeoX tokenizer. LayerLoop tuples give (K, L, R) and use final checkpoints. Original public-model data, optimization, and computation differ from ours, so these rows provide scale context rather than an architecture-controlled comparison.

Model	Params (M)	Tokens (B)	Tpp	NLL ↓	B_{suite} ↑
LayerLoop-608 $(16, 4, 4)$	434.2	4.342	10.00	2.841	71.23
LayerLoop-768 $(10, 4, 2)$	455.5	4.556	10.00	2.763	67.53
LayerLoop-1024 $(5, 4, 2)$	456.3	4.563	10.00	2.774	62.86
Pythia-410M	405.3	299.893	739.87	2.397	66.59
Pythia-1B	1,011.8	299.893	296.40	2.230	62.57
Pythia-1.4B	1,414.6	299.893	211.99	2.154	67.71
Pythia-2.8B	2,775.2	299.893	108.06	2.057	69.40
Pythia-6.9B	6,857.3	299.893	43.73	1.949	72.81
OLMo-7B	6,888.1	2,500.000	362.95	2.057	72.27

G.1 Processed tokens at the first observed score crossing

To expose the processed-token contrast directly, Table 15 reports the earliest evaluated Pythia checkpoint that reaches each score threshold. Let f identify one of the five evaluated Pythia model sizes and let $b \in [0, 100]$ be a suite-score threshold. Among that size's evaluated checkpoints, $T_f^{\text{first}}(b)$ is the smallest processed-token count with $B_{\text{suite}} \geq b$; it is unreported if none qualifies. The table's token multiple is

$$\rho_f(b) = \frac{T_f^{\text{first}}(b)}{T_{\text{LL,final}}}, \qquad T_{\text{LL,final}} = 4{,}341{,}694{,}464. \tag{8}$$

Here $T_{\text{LL,final}}$ is the processed-token count of the main LayerLoop final checkpoint. Thus, a ratio above one means the qualifying public checkpoint processed more tokens than the final LayerLoop model. At LayerLoop's final score, the observed crossings require 20.97–100.66 billion tokens across the five Pythia sizes, or 4.8–23.2 times LayerLoop's processed-token count. The crossing budget is not monotone across model sizes. Scores also fluctuate along each training trajectory: Pythia-6.9B scores 70.47 at 20.97 billion tokens and then dips before first exceeding the 71.23 threshold at 83.89 billion tokens.

Table 15: Earliest evaluated public Pythia checkpoint reaching each suite-score threshold. Tok. is processed tokens in billions; ×LL is $\rho_f(b)$ from Eq. (8). Dashes mean no evaluated checkpoint reached the threshold. Crossings use the full evaluated grid, which is denser than the 12-checkpoint anchor ladder. Because scores are non-monotonic, a crossing is the first observed point at or above a threshold, not a sustained level.

Threshold	Pythia-410M		Pythia-1B		Pythia-1.4B		Pythia-2.8B		Pythia-6.9B	
	Tok.	×LL	Tok.	×LL	Tok.	×LL	Tok.	×LL	Tok.	×LL
30	4.19	1.0×	2.10	0.5×	4.19	1.0×	4.19	1.0×	10.49	2.4×
35	6.29	1.4×	4.19	1.0×	4.19	1.0×	4.19	1.0×	10.49	2.4×
40	6.29	1.4×	6.29	1.4×	4.19	1.0×	4.19	1.0×	10.49	2.4×
45	6.29	1.4×	6.29	1.4×	6.29	1.4×	8.39	1.9×	10.49	2.4×
50	8.39	1.9×	6.29	1.4×	6.29	1.4×	8.39	1.9×	10.49	2.4×
55	12.58	2.9×	8.39	1.9×	6.29	1.4×	8.39	1.9×	20.97	4.8×
60	25.17	5.8×	8.39	1.9×	8.39	1.9×	8.39	1.9×	20.97	4.8×
65	27.26	6.3×	8.39	1.9×	16.78	3.9×	8.39	1.9×	20.97	4.8×
70	48.23	11.1×	16.78	3.9×	20.97	4.8×	20.97	4.8×	20.97	4.8×
71.23 (ours)	56.62	13.0×	100.66	23.2×	20.97	4.8×	20.97	4.8×	83.89	19.3×
72.5	58.72	13.5×	100.66	23.2×	20.97	4.8×	25.17	5.8×	83.89	19.3×
75	102.76	23.7×	100.66	23.2×	–	–	25.17	5.8×	83.89	19.3×
77.5	102.76	23.7×	–	–	–	–	184.55	42.5×	–	–
80	109.05	25.1×	–	–	–	–	–	–	–	–

H TOKENIZER-NATIVE 7B CONTEXT ON RAW TEXT TASKS

This appendix supplements the public-model scale context in Appendix G by asking which raw diagnostic comparisons remain interpretable when models use different tokenizers. We first state the comparison protocol, then report tasks where small LayerLoop models remain competitive, tasks where the public models lead, and a same-tokenizer training trajectory. The tokenizer-native Llama-2-7B, Qwen2.5-7B, and Gemma-7B runs provide model-scale context. Each model completed all 24 diagnostics after the same decoded source text was retokenized with its native tokenizer. Because token-boundary-sensitive components prevent direct comparison with the common-tokenizer B_{suite} rows, we report only raw results on decoded-text tasks for which the prompt, candidate meaning, and answer criterion are shared across models. These comparisons remain descriptive: likelihood-ranked candidates retain some residual tokenization sensitivity, and no result in this section estimates variation beyond the fixed prompts and checkpoints. The three LayerLoop final checkpoints are identified by hidden width: width 608 uses $(K, L, R) = (16, 4, 4)$, width 768 uses $(10, 4, 2)$, and width 1024 uses $(5, 4, 2)$. All subsequent width labels refer to these fixed configurations.

To compare results whose prompt and answer meaning are shared across tokenizers, we retain source-byte-normalized loss and named task accuracies. Table 16 records the inclusion rule. From the comparisons below we omit native-tokenizer NLL, token-level tail losses, token-level repetition measures, answer loss, the fixed-token-ID controlled synthetic task, and all expected calibration error (ECE) values. Bits per byte is retained because it normalizes source-text loss by the number of bytes rather than by the number of model-specific tokens. Exact-match and accuracy metrics are reported at their named task level. None of these restrictions changes the main LayerLoop–dense, matrix, scaling, or Pythia comparisons, all of which use the common GPT-NeoX tokenizer; they apply only to the tokenizer-native public-model context.

Table 16: Protocol used for tokenizer-native public-model context. "Retain" means suitable for descriptive raw-task comparison, not tokenizer invariance or a statistically powered benchmark.

Treatment	Measures	Reason
Retain	Bits per byte; decoded-text exact match; named task and subtask accuracy	Common source text and answer semantics; byte normalization avoids per-token NLL units
Omit	Native-tokenizer NLL; per-token tail loss; long/short loss gain; answer loss	Magnitude changes with token boundaries and the number of tokens per answer
Omit	Fixed-ID controlled synthetic accuracy	The synthetic generator assigns task roles to fixed vocabulary IDs, whose meanings differ across tokenizers
Omit	Token-run length and sampled token repetition	Token runs and token repetition rates change under retokenization
Omit	Choice, code, copy-induction, and order ECE	Changing answer segmentation can change token-level confidence.

H.1 Where small LayerLoop checkpoints remain competitive

Table 17 reports every decoded-text subtask on which at least one of the three LayerLoop checkpoints reaches or exceeds at least one tokenizer-native 7B reference. This is an explicit inclusion rule rather than a new score. The clearest pattern is localized rather than uniform. The width-768 checkpoint is competitive on cloze completion, counterfactual updates, reading, and long needle retrieval. The width-1024 checkpoint is competitive on order sensitivity. The width-608 checkpoint reaches the 7B range on several science, semantic, and entity slices. The latter slices are useful qualitative diagnostics, but a score of 1.0 there means correct answers, not evidence of population-level superiority.

Table 17: Raw decoded-text tasks on which the best LayerLoop checkpoint matches or exceeds at least one tokenizer-native 7B model. n is the number of examples in the full task or named subtask. The best LayerLoop checkpoint is selected independently in each row. Higher is better; these rows satisfy the stated inclusion rule, not necessarily a win against every public model.

Task	n	Best LayerLoop checkpoint	LayerLoop	Gemma-7B	Llama-2-7B	Qwen2.5-7B
Cloze completion	72	all three	0.917	1.000	0.750	1.000
Minimal-pair discrimination	72	all three	1.000	1.000	1.000	1.000
Counterfactual	48	width 768	0.667	0.750	0.417	0.917
Order sensitivity	72	width 1024	0.556	0.556	0.222	0.667
Reading comprehension	72	width 768 / width 608	0.667	0.917	0.667	0.833
Reading: single fact	27	width 768	1.000	1.000	0.667	1.000
Knowledge recall	96	width 608 / width 1024	0.917	0.917	1.000	1.000
Temporal/causal reasoning	72	width 768	0.778	1.000	0.778	1.000
Needle retrieval at 2048 tokens	36	width 768 / width 1024	1.000	1.000	1.000	1.000
Benchmark: science	54	width 608	1.000	1.000	0.667	1.000
Analogy: semantic	54	width 768 / width 608	1.000	1.000	0.667	1.000
Entity tracking: ownership	54	width 608	1.000	1.000	0.667	0.667

These selected slices describe where the small LayerLoop checkpoints can match at least one larger public model on the evaluated prompts. They do not establish that LayerLoop is globally stronger than a 7B model. In particular, several rows are ties, and Qwen2.5-7B or Gemma-7B remains strongest on most full task sections reported here. The strength and weakness table rules are not mutually exclusive: a task can appear in both when LayerLoop ties one public model but every public model matches or exceeds it, as in Knowledge recall.

H.2 Where the 7B models remain stronger

The same evaluation also identifies clear limitations (Table 18). All three public 7B models achieve substantially lower bits per byte and stronger code, arithmetic, distractor robustness, and text exact match. The gap is also broad on few-shot adaptation, symbolic reasoning, truthfulness, and definition recall. These outcomes are compatible with the much larger public models' parameter counts and training corpora and prevent interpreting the isolated wins above as general 7B-level capability.

Table 18: Representative raw tasks on which every tokenizer-native 7B model matches or exceeds the best LayerLoop checkpoint. The best LayerLoop checkpoint is selected independently in each row. Bits per byte is lower-better; all other rows are higher-better accuracies.

Task	n	Best LayerLoop checkpoint	LayerLoop	Gemma-7B	Llama-2-7B	Qwen2.5-7B
Bits per byte ↓	–	width 768	1.024	0.761	0.755	0.727
Code reasoning	72	width 768	0.667	1.000	1.000	1.000
Math word problems	72	width 768 / width 1024	0.083	0.750	0.500	0.500
Distractor retrieval	108	width 768	0.333	1.000	1.000	1.000
Structured-text exact match	192	width 768 / width 608	0.062	0.292	0.312	0.667
Knowledge recall	96	width 608 / width 1024	0.917	0.917	1.000	1.000
Few-shot pattern following	60	width 608 / width 1024	0.600	1.000	0.867	1.000
Symbolic reasoning	96	width 1024	0.375	0.875	0.500	0.875
Truthfulness	72	width 608	0.583	0.833	0.667	0.750
Definition recall	72	width 768 / width 608	0.556	0.889	0.667	1.000
Needle retrieval, all lengths	108	width 768	0.889	1.000	1.000	1.000

H.3 Same-tokenizer Pythia trajectory on representative raw tasks

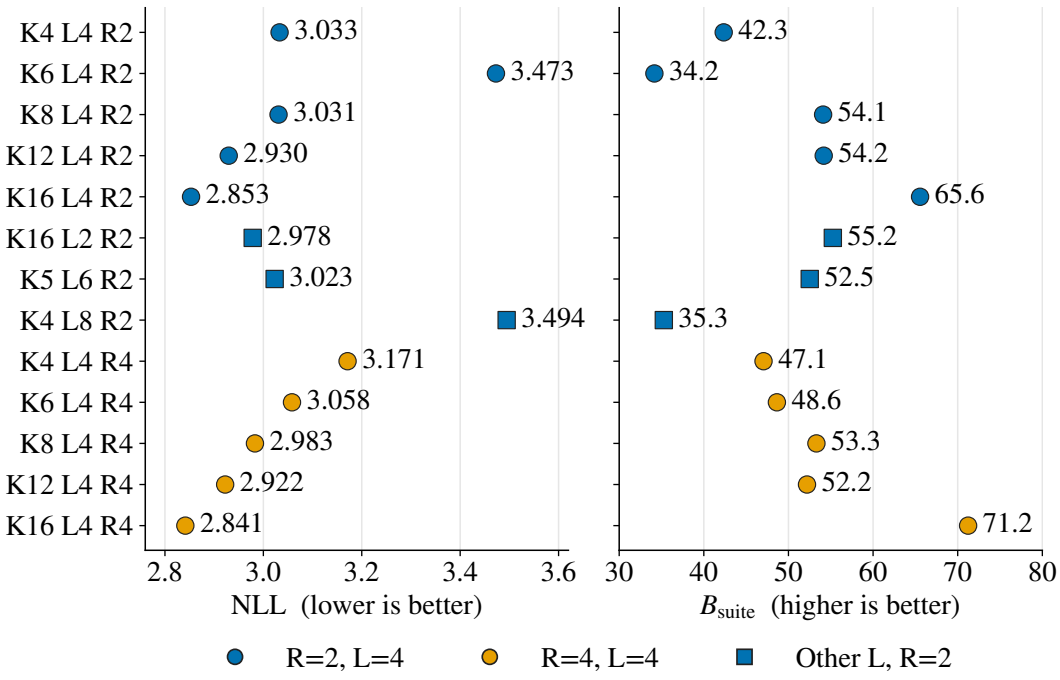
Pythia uses the same GPT-NeoX tokenizer as LayerLoop, so the richer common-tokenizer comparison can retain both bits per byte and the original B_{suite} elsewhere in the paper. Table 19 adds raw task context without changing that compact-model comparison. At roughly 4.3–4.6B processed tokens, the LayerLoop rows are already stronger than Pythia-410M step2000 on cloze, counterfactual, reading, code, and needle retrieval. Several of these advantages persist against much later Pythia-410M checkpoints, whereas Pythia's language-model fit improves with additional training and its code and retrieval accuracies follow different, not always monotone, trajectories. Pythia-6.9B provides the strongest language-model and code reference.

Table 19: Representative raw metrics along the same-tokenizer Pythia trajectory. BPB is bits per byte (lower is better); all remaining columns are accuracy (higher is better). Needle reports accuracy over 108 examples pooled across all tested lengths. Bold marks the highest value per accuracy column and the lowest bits-per-byte value. "s2k" denotes step2000, and analogously for later revisions.

Model	Tokens (B)	BPB ↓	Cloze	Counterf.	Order	Reading	Code	Math	Needle all	Truth
LayerLoop-768 $(10, 4, 2)$	4.556	1.024	**0.917**	**0.667**	0.333	**0.667**	0.667	0.083	0.889	0.500
LayerLoop-608 $(16, 4, 4)$	4.342	1.035	**0.917**	0.583	0.222	**0.667**	0.444	0.000	0.667	**0.583**
LayerLoop-1024 $(5, 4, 2)$	4.563	1.028	**0.917**	0.583	0.556	0.583	0.444	0.083	0.667	0.500
Pythia-410M, s2k	4.194	1.221	0.750	0.500	**0.667**	0.333	0.333	0.083	0.278	0.500
Pythia-410M, s8k	16.777	1.004	0.833	0.500	0.222	**0.667**	0.444	**0.167**	0.889	0.417
Pythia-410M, s56k	117.441	0.914	0.750	0.583	0.333	**0.667**	0.778	0.083	**1.000**	0.500
Pythia-410M, main	299.893	0.888	0.750	0.500	**0.667**	0.583	0.556	0.000	**1.000**	0.500
Pythia-6.9B, main	299.893	**0.723**	0.833	**0.667**	0.444	0.583	**1.000**	0.083	**1.000**	0.500

I Width-608 Depth and Granularity Results

This appendix extends the matched depth and granularity comparisons in Sections 4.2 and 4.4. Figure 10 visualizes the archived width-608 designs and shows how the number of stages, stage granularity, and repeat count shape the two reported summaries. The highest-scoring model on the diagnostic suite is $(K, L, R) = (16, 4, 4)$; NLL and aggregate score vary non-monotonically across the three axes. This is an inventory across budgets, not one fixed-budget comparison: Section 4.2 selects the matched-depth comparisons, and Section 4.4 compares each depth with its dense counterpart.

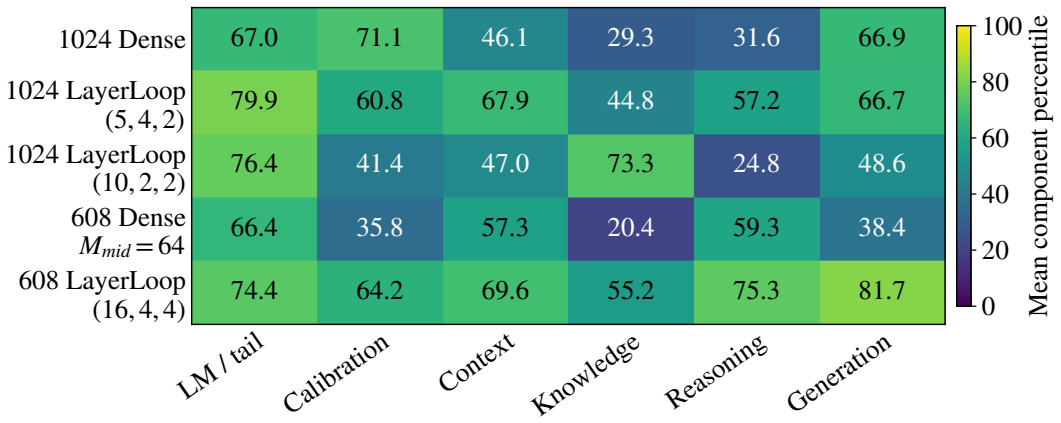


(a) Held-out NLL (b) Diagnostic suite

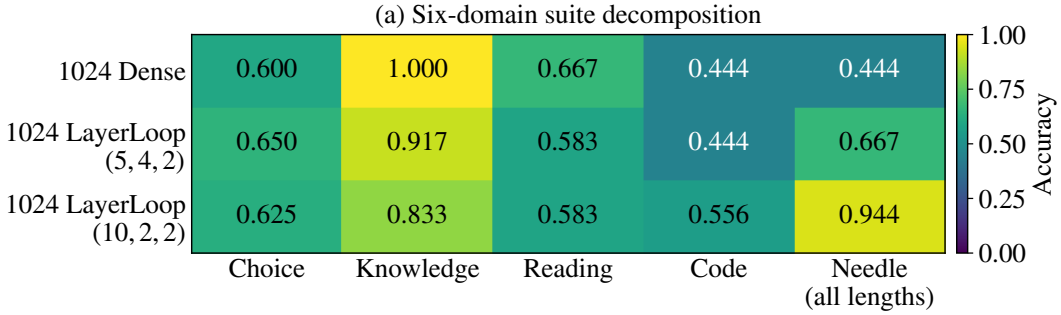
Figure 10: Width-608 results across stage count K, blocks per stage L, and repeat count R. Blue marks use $R = 2$ and orange marks use $R = 4$; squares identify the three non-$L = 4$ designs. The panels share configuration rows and order. The configurations span different parameter and processed-token budgets, so this is an inventory rather than one controlled comparison. The $K5, L6, R2$ model is a smaller-budget context row.

J Additional Capability Profiles

This appendix examines the domain and raw-task results behind Section 4.4. The top panel asks which domains contribute to the aggregate advantage; the bottom checks named raw accuracies at width 1024 (Figure 11). At width 1024, the four-block-stage model $(5, 4, 2)$ leads Dense in four of six domains. The two-block-stage model $(10, 2, 2)$ leads the other two models on code and needle retrieval. The contrasting rankings show why aggregate and task-specific measures are both informative.



(a) Six-domain suite decomposition



(b) Width-1024 raw accuracies

Figure 11: Stage designs differ across domains and named raw tasks. Top: six-domain decomposition of the 24-component suite for the width-1024 comparison and the matched width-608 dense–LayerLoop pair; top-panel cells are equal-weight means of percentile ranks for components assigned to each domain (Table 4). Bottom: named raw accuracies for the width-1024 comparison, using 160 choice, 96 knowledge, 72 reading, 72 code, and 108 needle examples. Higher is better in both panels; model tuples give (K, L, R). Percentiles use a 0–100 scale; raw accuracies are fractions of examples on a 0–1 scale. The panels do not share axes or scales.

K Stage-Boundary Residual Training Ablation

This appendix reports a short-budget comparison of the stage-boundary residual, separate from the complete-design comparisons in Section 4.2. Table 20 gives the observed values for the width-608 $(K, L, R) = (16, 4, 4)$ model. After all repeats, the control adds the stage input to the repeated computation's output and applies root-mean-square normalization to the sum. The ablated variant normalizes only the repeated computation's output. Both use the prepared Pile corpus, the same 434.156M stored-parameter architecture, AdamW schedule, bfloat16 arithmetic, two-device distributed training, 65,536 tokens per update, and a 300M-token budget. The frozen configuration files differ only in the residual toggle and their assigned initialization seeds.

Table 20: Matched 300M-token training ablation of the stage-boundary residual. Validation NLL is computed on the same internal held-out stream; lower is better. One independently initialized run is available for each variant, so the table records the observed contrast in this configuration but does not estimate across-seed variability.

Variant	Residual	Seed	Tokens	Best NLL ↓	Final NLL ↓
LayerLoop control	Present	401	300M	**3.767**	**3.883**
Without stage residual	Removed	402	300M	4.128	4.211

The best-checkpoint difference is 0.361 NLL and the final-endpoint difference is 0.328 NLL. Both runs complete the specified budget, and retaining the stage-boundary residual yields lower loss at

both reported checkpoints. This short-budget, one-run-per-variant comparison does not separately isolate how stage granularity changes quality or replace the controlled granularity comparisons in the main text.

L Parameter-Sensitivity, Hidden-Unit, and State-Change Diagnostics

This appendix defines the measurements behind Section 4.5: parameter sensitivity, hidden-unit activation frequency, and hidden-state change across the executed path. The sensitivity and activation comparisons pair LayerLoop with a dense model at each of widths 608 and 1024, matching stored parameters and processed tokens.

Parameter sensitivity. To measure whether loss sensitivity is concentrated in a small fraction of stored parameters, let $p \in \{1, \ldots, P_{\text{stored}}\}$ index stored scalar parameter w_p. We partition the fixed held-out window of 65,536 Pile tokens into $B = 32$ microbatches. Each microbatch contains one 2,048-token sequence. Let $v \in \{1, \ldots, B\}$ index microbatches and let $g_{p,v} = \partial \mathcal{L}_v / \partial w_p$ denote the gradient of mean microbatch NLL with respect to stored scalar parameter w_p. We differentiate the complete forward computation, including all repeated uses, without updating the parameters. For a small fractional perturbation of w_p, the first-order loss change is proportional to $w_p g_{p,v}$. We average its absolute value across microbatches:

$$q_p = \frac{1}{B} \sum_{v=1}^{B} |w_p g_{p,v}|. \tag{9}$$

We report the Gini coefficient (Hurley & Rickard, 2009) over all stored-parameter scores. Writing $N = P_{\text{stored}}$ and $q_{(1)} \leq \cdots \leq q_{(N)}$ for the sorted scores, its value is $2 \sum_{p=1}^{N} p q_{(p)} / (N \sum_{p=1}^{N} q_{(p)}) - (N+1)/N$. The observed score sums are positive. A lower value indicates less concentration across parameters.

Hidden-unit activation. We also measure how often each stored MLP hidden unit is active. One event is the unit's activation at one token position during one execution of its stored block, so recurrent blocks contribute one event per token and repeat step. We call a unit low-activation when its absolute hidden-unit activation exceeds 10^{-3} on fewer than 1% of these token-step events. For each unit, we average the threshold indicator across the fixed-window positions and all executions of its stored block. The reported fraction averages the resulting low-activation indicators over stored MLP units, counting each unit once regardless of repeat count.

Figure 5 shows the same direction in both pairs: LayerLoop has less concentrated parameter sensitivity and a smaller fraction of low-activation units.

Figure 12 adds the fixed-depth granularity controls: whole-middle, four-block-stage LayerLoop, and blockwise recurrence at width 608. These three models each store 68 blocks, execute 260 block applications, repeat each reusable block four times, and process approximately 4.342 billion tokens. Their sensitivity concentrations occupy a narrow range despite their different stage-granularity values. Low-activation MLP fractions are 1.23% and 5.24% for the two dense models and 0% for the three width-608 granularity controls and the width-1024 LayerLoop model.

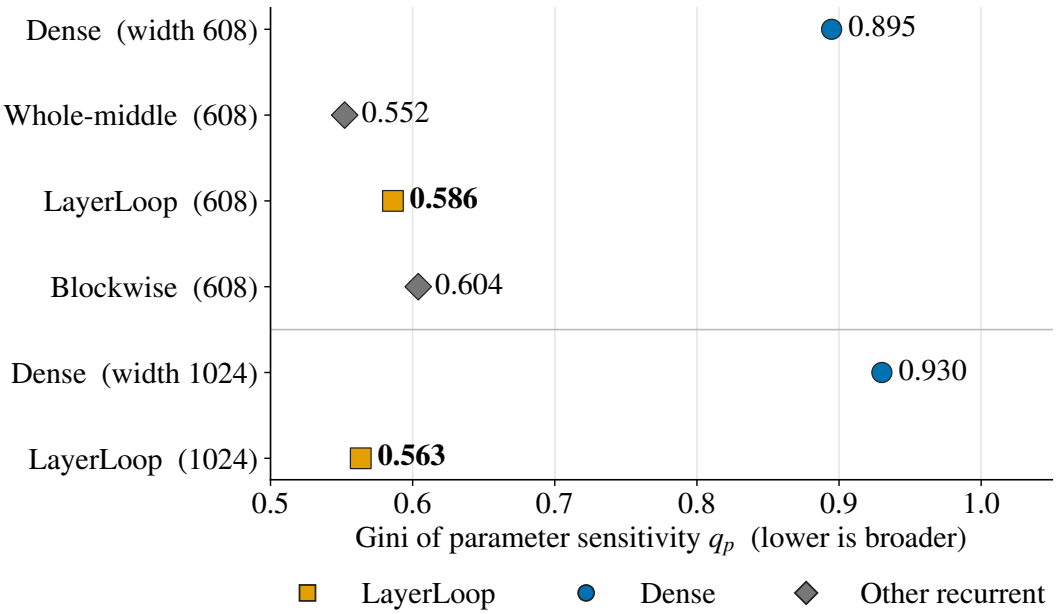


Figure 12: Parameter-sensitivity concentration across the two dense–LayerLoop pairs and the width-608 fixed-depth granularity controls. Lower Gini means broader participation. The width-608 whole-middle, four-block-stage LayerLoop, and blockwise controls each store 68 and execute 260 blocks; width-608 Dense executes 68. At width 1024, Dense stores and executes 24 blocks, while LayerLoop stores 24 and executes 44. The horizontal axis starts at 0.5 to resolve the observed values.

Hidden-state change along the path. For the width-608 pair, we measure the change produced by a block or a complete stage execution. Let $\vec{z}^{\text{in}}_{u,t}$ and $\vec{z}^{\text{out}}_{u,t}$ be the width-d input and output of measured computation unit u at token event t. We define

$$\delta_u = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \left\| \vec{z}^{\text{out}}_{u,t} - \vec{z}^{\text{in}}_{u,t} \right\|_2 , \tag{10}$$

where $\mathcal{T}$ contains all batch and token positions in eight held-out batches of four 2,048-token sequences, or 65,536 tokens in total. These are the same held-out windows used above; the parameter-sensitivity calculation splits them into 32 single-sequence microbatches. For the 68-block dense model, u identifies one Transformer block. We divide the mean δ_u over blocks 59–68 by that over blocks 1–10. For LayerLoop, $u = (i, r)$ identifies one complete stage-block-stack application for stage i and repeat r, measured after the repeat-conditioning vector and before the stage-boundary residual and stage-boundary normalization. We average $\delta_{i,r}$ over all 16 stages for each repeat index and divide repeat 4 by repeat 1. The resulting width-608 ratios are 0.0429 for Dense and 0.996 for LayerLoop. They summarize separate within-model axes and computation units and are not estimates of one shared cross-model ratio; Dense has no repeat-axis counterpart.

Under these processed-token budgets, the tested recurrent models have less concentrated parameter sensitivity, and LayerLoop continues to transform hidden states across repeats. These measurements characterize the internal computation of the tested checkpoints without attributing their quality differences to either diagnostic.

M Stage-Intervention Profiles Within and Across Checkpoints

This appendix specifies the stage-intervention-profile comparisons in Section 4.6. We measure the behavioral change caused by the stage-bypass intervention, apply stage-position detrending, and test whether the remaining profile identifies the same stage in another evaluation sample. The following paragraphs first analyze one fixed trained model; Section M.1 then applies the protocol to a separately trained checkpoint and compares the two checkpoints.

Checkpoint, inputs, and interventions. We use the final width-608 $(K, L, R) = (16, 4, 4)$ model, numbering stages 1–16 in computation order. Let $\mathcal{D}_a$, $a \in \{1, 2, 3\}$, denote the evaluation sets generated with sampling seeds 1337, 2027, and 4099. Each combines task prompts and candidate answers with sampled windows from the Pile internal holdout. Choice-task builders use two cases per group, distractor retrieval uses one case per bucket, and needle retrieval uses lengths 512 and 1024 with one case per insertion position. The prediction-loss component uses four batches of two 1,024-token sequences. The executable sampling entrypoint is `scripts/exp034_stage_functional_specialization.py`; it calls the task builders in `scripts/exp006_posttrain_eval.py` and records the full prompts, candidates, and per-case responses. Within each $\mathcal{D}_a$, every intervention uses the same inputs and candidate order as the intact model.

For input $\mathbf{H}_{i,0}$ to stage $i \in \{1, \ldots, 16\}$, the intact stage returns its repeated output plus $\mathbf{H}_{i,0}$, followed by stage-boundary normalization. Our primary intervention replaces the entire repeated computation, including all repeat-conditioning vectors, by a direct input pass-through. Stage i therefore returns $\mathtt{N}_i(2\mathbf{H}_{i,0})$. This leaves the input addition and learned stage-boundary normalization active. We change one stage's forward computation at a time and keep all trained parameters fixed.

Diagnostic response coordinates. A profile describes the pattern of behavioral changes rather than the mean rank B_{suite}. We use 22 raw response coordinates, not a subset formed by dropping two of its 24 components. Twenty coordinates are task-mean correct-versus-best-alternative score margins: analogy, benchmark-style choice, code, copy induction, counterfactual reasoning, definition recall, distractor retrieval, entity tracking, few-shot adaptation, forced choice, knowledge, math word problems, negation, natural-language inference, order, reading, symbolic reasoning, temporal/causal reasoning, truthfulness, and Winograd-style reference resolution. These margins subtract answer-only from prompted candidate log-score sums, then divide by the number of candidate tokens scored in the prompted setting. The margin is the correct candidate's adjusted score minus the highest incorrect-candidate score. Needle retrieval supplies an unadjusted mean margin, and raw NLL supplies the mean holdout prediction loss. The task names refer to the local diagnostic builders.

Let $\mu_c^{(a)}$ be the intact-model value for coordinate $c \in \{1, \ldots, 22\}$ on $\mathcal{D}_a$, and $\mu_{i,c}^{(a)}$ the corresponding value after applying the stage-bypass intervention to stage i. The signed degradation $\Delta_{i,c}^{(a)}$ is $\mu_{i,c}^{(a)} - \mu_c^{(a)}$ for NLL and $\mu_c^{(a)} - \mu_{i,c}^{(a)}$ for margins, so positive values indicate degradation. Each task margin averages its evaluated cases equally; NLL pools sampled token losses across sequences.

Profile construction. We separate stage-specific profile shape from the tendency of early or late stages to have larger absolute response changes. For each evaluation set and coordinate separately, we fit a least-squares line across the 16 stage positions. With fitted intercept $\hat{\beta}_{0,c}^{(a)}$ and slope $\hat{\beta}_{1,c}^{(a)}$, the residual harm is $Z_{i,c}^{(a)} = \Delta_{i,c}^{(a)} - \hat{\beta}_{0,c}^{(a)} - \hat{\beta}_{1,c}^{(a)} i$. We then center and scale the 22 coordinates within each stage:

$$V_{i,c}^{(a)} = \frac{Z_{i,c}^{(a)} - \bar{Z}_i^{(a)}}{\sigma_i^{(a)}}, \quad \bar{Z}_i^{(a)} = \frac{1}{22}\sum_{c=1}^{22} Z_{i,c}^{(a)}, \quad (\sigma_i^{(a)})^2 = \frac{1}{22}\sum_{c=1}^{22}(Z_{i,c}^{(a)} - \bar{Z}_i^{(a)})^2. \tag{11}$$

A divisor below 10^{-12} is replaced by one. The resulting column vector $\vec{V}_i^{(a)} = (V_{i,1}^{(a)}, \ldots, V_{i,22}^{(a)})^\top \in \mathbb{R}^{22}$ is the stage-intervention profile. Fitting and standardization are repeated independently for each $\mathcal{D}_a$; no fitted coefficients are shared across evaluation sets.

Cross-sample similarity and identification. For each pair $a < b$, we compare all stage-intervention profiles using cosine similarity:

$$C_{i,i'}^{(a,b)} = \frac{(\vec{V}_i^{(a)})^\top \vec{V}_{i'}^{(b)}}{\|\vec{V}_i^{(a)}\|_2 \, \|\vec{V}_{i'}^{(b)}\|_2}, \qquad i, i' \in \{1, \ldots, 16\}.$$

As in profile scaling, norms below 10^{-12} are replaced by one. The scalar $C_{i,i'}^{(a,b)}$ compares vector directions. The 16×16 matrix of these values has corresponding stages on its diagonal. Pooling the three pairs gives 48 same-stage values and 720 different-stage values, with medians 0.6608 and -0.0021.

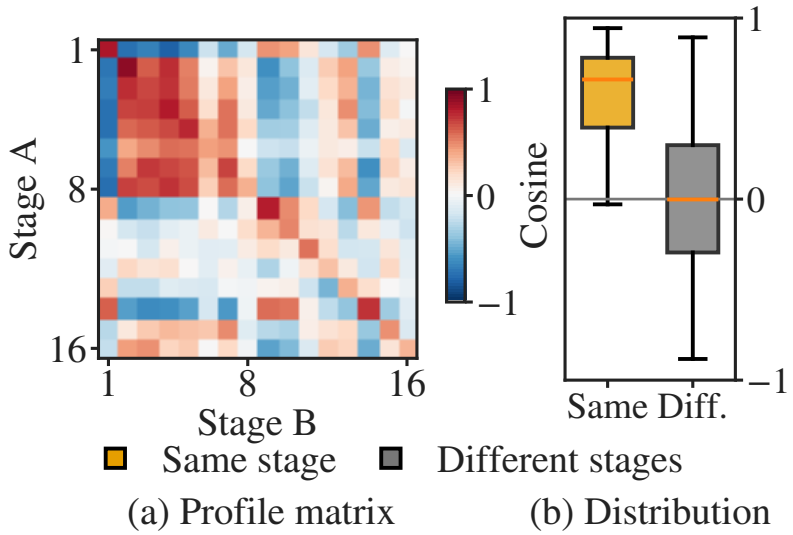


Figure 13: Stage-intervention profiles across three evaluation-set pairs. (a) Mean cosine similarity by stage position in samples A and B; (b) distributions of 48 same-stage and 720 different-stage similarities after stage-position detrending. Boxes show quartiles and medians, whiskers reach up to 1.5 interquartile ranges, and outliers are omitted. The panels have different axes and scales.

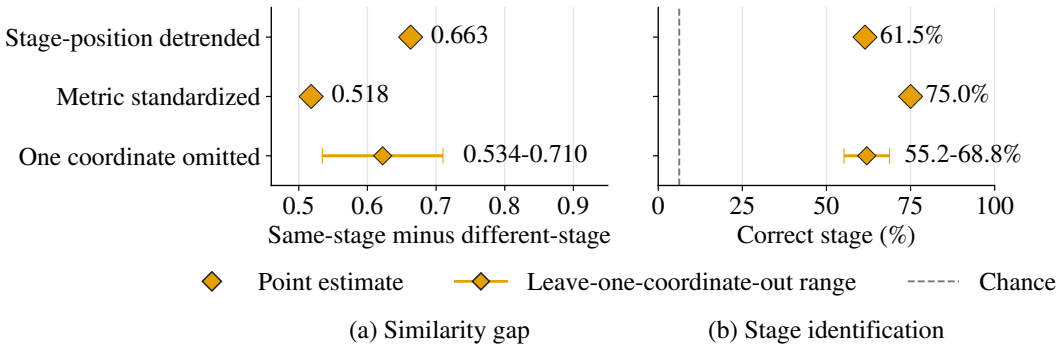
For identification, each stage-intervention profile retrieves the most similar stage in the other set. Each set pair contributes 16 queries in each direction, giving 96 queries in total; 59 identify the correct stage (61.46%), compared with chance $1/16 = 6.25\%$. A 10,000-permutation test independently reassigns comparison-stage labels within each set pair and recomputes the mean diagonal similarity, then averages across the three pairs. With the observed statistic included among the permutation trials in both the numerator and denominator, the resulting permutation-test value is $p = 1/10{,}001$. These queries share a checkpoint and evaluation sets; the test randomizes stage labels and does not estimate variability across training runs.

Robustness and replacement controls. To check sensitivity to coordinate scales, we additionally standardize each residualized coordinate across stages before applying the same within-stage standardization. Identification rises to 75.0%. Removing each coordinate in turn also retains the same-stage separation (Figure 14).

To check the role of retained stage-boundary normalization, a direct identity control returns $\mathbf{H}_{i,0}$ without normalization. Its 16 stagewise NLL changes on the seed-1337 inputs have Pearson correlation 0.9999 with the primary intervention, showing similar stagewise loss patterns. To test interchangeability, a second control performs complete-stage replacement by replacing one entire trained stage module with another among stages 1, 6, 11, and 16. The source module includes its blocks, repeat-conditioning vectors, residual-scale parameters, and stage-boundary normalization; the target position retains the same stage-boundary residual operation. All 12 directed replacements increase mean token NLL by 0.075–10.478, with median 0.751, on the seed-1337 inputs. This establishes non-interchangeability of those complete stages at the tested positions; the mixed changes across tasks do not assign exclusive semantic functions to individual stages.

M.1 Independent short-budget checkpoint and cross-checkpoint alignment

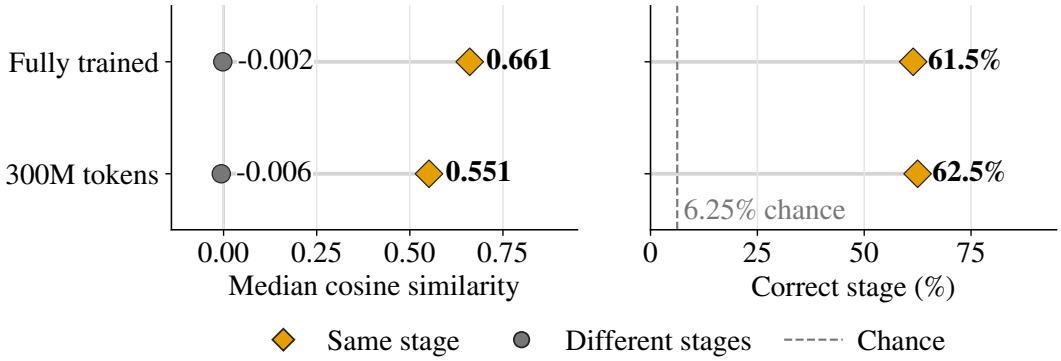
We apply the same 22-coordinate intervention protocol to an independently trained $(K, L, R) = (16, 4, 4)$ width-608 checkpoint. This checkpoint uses the same prepared Pile corpus, tokenizer, architecture, and optimizer family, but it has training seed 401, stops at 300,023,808 processed tokens, and uses 65,536 tokens per optimizer update. It is the residual-retaining control run in Appendix K. The fully trained checkpoint processes 4.342 billion tokens and uses 196,608 tokens per update. Thus, the comparison changes training run, budget, and update batch together; it is not an isolated seed comparison.



(a) Similarity gap (b) Stage identification

Figure 14: Stage differentiation persists under changes to profile construction. Left: the gap between same-stage and different-stage median cosine similarity. Right: identification over the 96 cross-set queries; the dashed line marks 6.25% chance. The first two rows show the stage-position-detrended and metric-standardized point estimates. Error bars on the third row span the 22 leave-one-coordinate-out analyses. The panels share analysis rows and order.

For each checkpoint separately, we use evaluation-set seeds 1337, 2027, and 4099, apply the same stage-bypass intervention to all 16 stages, apply coordinate-wise stage-position detrending, and standardize each stage-intervention profile. The short-budget checkpoint has same-stage and different-stage median similarities of 0.5515 and -0.0062, respectively. Bidirectional identification is 60 of 96 queries (62.5%). The 10,000-permutation test of mean diagonal similarity gives $p = 1/10{,}001$.



(a) Intervention-profile similarity (b) Stage identification

Figure 15: Stage-intervention profiles recur across sampled evaluation sets in both checkpoints. Left: same-stage and different-stage median cosine similarities after stage-position detrending. Right: correct-stage identification over 96 cross-set queries; dashed line is 6.25% chance. The panels share checkpoint rows and order. For each checkpoint, $p = 1/10{,}001$ tests mean same-stage similarity under permuted stage labels, not identification accuracy.

We then compare the two checkpoints directly. For each shared evaluation seed, every stage-intervention profile from the fully trained checkpoint is compared with every profile from the short-budget checkpoint, giving three 16×16 matrices. The diagonal supplies 48 same-position similarities and the off-diagonal supplies 720 different-position similarities. Their medians are 0.0077 and -0.0038, respectively. Bidirectional retrieval identifies the same stage position in 8 of 96 queries (8.33%), against 6.25% chance. Reassigning stage labels independently within each matrix for 10,000 permutations gives $p = 0.296$ for the observed mean diagonal similarity.

The short-budget result independently reproduces within-checkpoint stage differentiation. In the direct comparison, we do not detect reliable same-position profile alignment between these two checkpoints. Seed, training budget, and update batch change together, so the cross-checkpoint comparison neither identifies which change matters nor establishes training-condition specificity in general.

N LIMITATIONS

The comparisons use one prepared corpus, sub-billion-parameter models, and one training run per architecture–configuration pair; same-budget retraining variability is unmeasured. The 24-component suite combines reference-relative percentiles from diagnostic tasks. Its ranking is not a general-capability ranking: public-benchmark leaders differ by task, and all evaluated checkpoints remain near chance on zero-shot MMLU. Prediction loss also has a different ordering: blockwise recurrence has the lowest NLL in the four-repeat controlled comparison, and Parcae has lower NLL than four-block LayerLoop among the external recurrent baselines. The compact-model NLL estimates are non-paired samples from the same holdout, so small numerical differences are descriptive rather than paired-sample effects.

Matching stored parameters and processed tokens does not match training computation or executed depth. In the width-608 and width-1024 Dense–LayerLoop pairs, LayerLoop applies 3.82 and 1.83 times as many blocks. With batch size one and full-context recomputation on an NVIDIA L40, median prefill latency is 3.86 and 1.80 times higher and median per-token decode latency is 4.01 and 1.82 times higher, respectively. These measurements do not characterize cached or optimized serving; Appendix A gives the timing protocol and absolute latencies (Figure 6).

Broader parameter participation is measured by first-order sensitivity concentration and the fraction of low-activation multilayer-perceptron units, not by a causal test of the suite gains. The trained stage-response profiles distinguish positions within each tested checkpoint but do not assign exclusive semantic functions. The two profile checkpoints differ in seed, processed tokens, and update batch, so their comparison does not isolate one training condition or establish stable roles across runs. Complete-stage swaps test four selected stages, not every possible replacement.

O AI USE STATEMENT

We used generative AI tools to assist with literature organization, experimental and figure-generation code, data analysis, and manuscript drafting and revision. The authors reviewed all AI-assisted text and code, verified reported measurements against saved result files, and compiled and visually inspected the final paper. We take responsibility for the complete content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.