THE READOUT BLIND SPOT IN LOOPED LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped language models iteratively apply the same transformer block, which enables us to vary computational depth at inference time. To ensure intermediate loops produce useful predictions, these models are usually trained with dense supervision applying a cross-entropy loss at every loop. In this work, we study a supervision failure that arises from this practice, which we term the *readout blind spot*. Because standard output readouts normalize the hidden state before prediction, they effectively mask the state's magnitude from the local loss gradient. Meanwhile, pre-norm residual connections continuously accumulate scale. As a result, dense supervision forces intermediate states to become predictive but entirely fails to regulate their residual norm, allowing it to grow uncontrollably across loops. Through ablations at 44M and 129M, and a matched 1.4B scaling study, we demonstrate that stabilizing looped models requires actively decoupling exit supervision from scale control. Exposing the scale to the loss (via raw readouts or norm penalties) or resetting it within the loop (via inter-loop normalization) successfully bounds the residual norm. Every form of scale control improved perplexity in our small-model recipes and at 1.4B parameters, the blind spot hold in norm readout.

1 INTRODUCTION

By iteratively reusing the same transformer block, looped language models allow computational depth to vary independently of parameter count (Dehghani et al., 2019; Geiping et al., 2025; Zhu et al., 2025). This architectural choice offers a natural mechanism for adaptive computation, where easy tokens can exit early and difficult ones can be processed further. However, realizing this promise requires managing the hidden state across iterations. Because every later loop starts from the state produced by the earlier ones, the hidden state takes on a dual role: it serves both as an interface for immediate prediction and as the continuous memory for the recurrence. This dual role creates a subtle but profound supervision problem: *when we apply a loss at every loop, what exactly are we controlling in the recurrent state?*

The intuitive approach to training these intermediate exits is dense supervision, where a cross-entropy (CE) loss is applied at every loop to provide a direct learning signal. While this successfully trains the exits to become predictive, it leaves the internal dynamics of the residual stream largely unconstrained. For example, in 44M and 129M models trained with dense supervision and standard RMSNorm prediction heads, the exits produce accurate predictions, yet the mean final-loop hidden state norms reach roughly 39,000 and 56,000. Removing the output normalizer brings these norms down to 44 and 77. Per-loop supervision makes the intermediate states usable, but it does not intrinsically prevent the recurrent norm from drifting (Figure 1).

This effect arises from a structural misalignment between the prediction readout and the recurrent update. Because standard output readouts normalize the hidden state before calculating logits, they strip away the vector's magnitude, leaving the immediate CE loss with almost zero gradient in the radial direction. However, the pre-norm recurrence still receives the unnormalized state. If the learned residual updates have a positive radial component, the scale will continue to accumulate. We term this misalignment between what the prediction head sees and what the recurrence carries the **readout blind spot**.

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

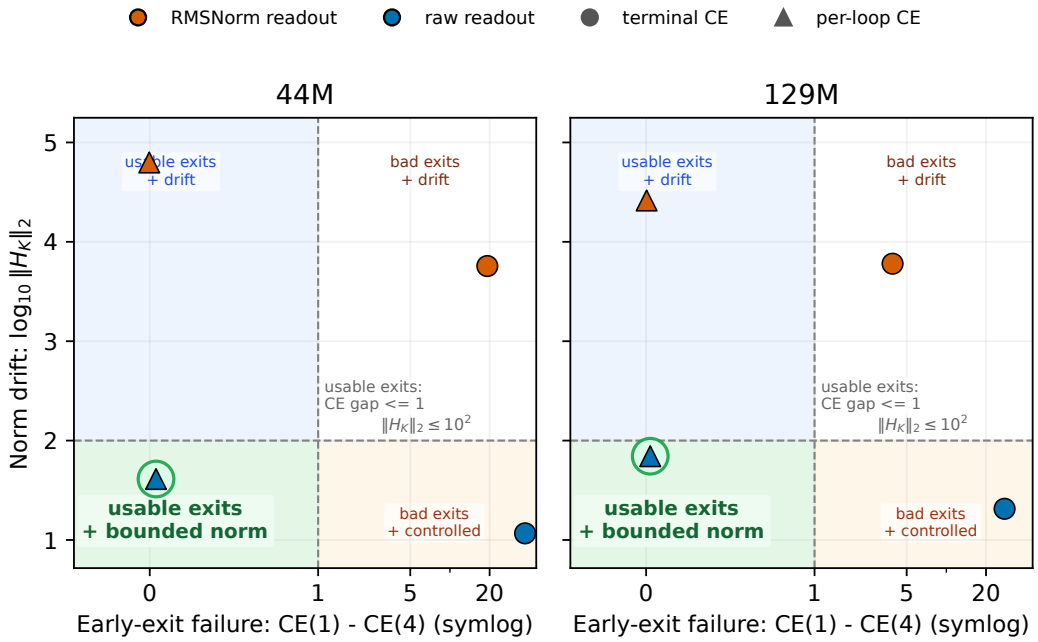

Figure 1: **Exit training and scale control are independent objectives.** Models are positioned by early-exit usability (the cross-entropy gap between loops 1 and 4, horizontal axis, symlog scale) and residual stability (the mean final-loop token norm, vertical axis, log scale). Per-loop supervision narrows the exit gap, making intermediate loops usable for prediction, while scale-visible interventions (raw readouts and norm penalties) pull the models downward by bounded the norm. Results reflect 44M and 129M models trained with our small-model recipe.

Observation 1.1 (Readout blind spot, informal)**.** *Because a normalized readout masks the hidden state's magnitude from the loss, per-loop cross entropy provides almost no gradient to regulate scale. Meanwhile, pre-norm recurrence continuously updates and accumulates that scale. Dense supervision therefore trains the prediction exits, but fails to control the scale of the recurrent state.*

We validate this mechanism through targeted interventions on both the loss and the recurrent path. By exposing the scale to the loss (via an unnormalized readout or an explicit norm penalty) or by removing accumulated scale from the recurrence (via inter-loop normalization), we suppress norm growth across 44M, 129M, and 1.4B models. Furthermore, when we causally clamp the accumulated scale after the first loop during inference, the evaluation cross-entropy barely changes, confirming that the scale growth is predictively neutral.

Our contributions are:

1. **Identifying the readout blind spot.** We analyze how the readout hides scale from each exit's immediate loss while the recurrence keeps changing it, and prove that the gradient later losses send back to the scale decays as $1/s_k$ (Section 2).
2. **Evidence across scales and recipes.** We measure norm growth in 44M, 129M, and 1.4B models under two training recipes, show that raw readouts, norm penalties, and inter-loop normalization each keep the norm bounded, and show that clamping the built-up scale at inference barely changes the loss (Sections 4–6).
3. **Exits, scale, and depth are separate.** Whether bounding the norm improves perplexity depends on the training recipe: every scale-control method lowers perplexity by 8–11% under the small-model recipe, while under the 1.4B recipe the loss-side methods never lower it by more than 1% in any run. Exit usability, scale control, and the gain from extra loops therefore have to be measured separately (Sections 5 and 7).

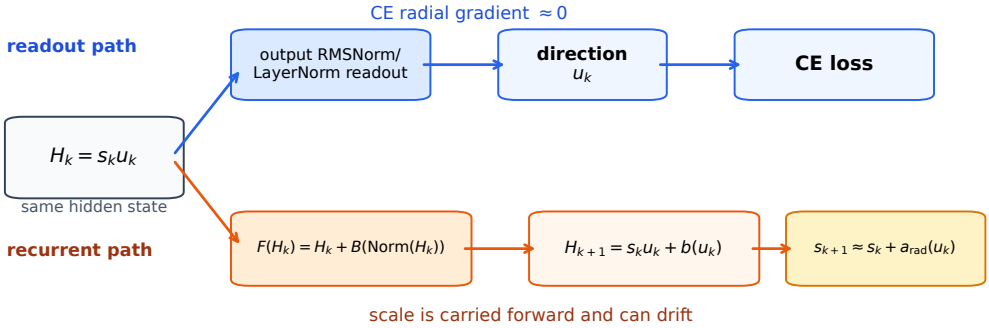

Figure 2: **What the readout hides and what the recurrence carries.** The same hidden state $H_k = s_k u_k$ is the input to the readout and to the next loop. Along the readout path, normalization removes scale, so the immediate CE loss gives no gradient on it. Along the recurrent path, the pre-norm residual update still carries the full state H_k, so scale keeps accumulating, by roughly a constant amount per loop (Lemma 2). Here Norm denotes the pre-sublayer normalizer and B denotes the residual branch applied to the normalized state.

2 Mechanism: The Readout Blind Spot Masks the Norm Gradient

At each loop k, a looped model uses its hidden state H_k in two ways. It reads out a prediction for the next token, and it feeds H_k back into the shared block to produce the next state:

$$H_{k+1} = F_\theta(H_k), \qquad z_k = W_{\text{out}}\, r(H_k), \qquad \mathcal{L}_k = \text{CE}(z_k, y). \tag{1}$$

Here F_θ is the shared transformer block, y is the next-token target, and $\mathcal{L}_k$ is the cross-entropy loss at loop k. We call the map from H_k to the logits z_k the *readout* (often called the LM head). It first applies a normalization r (RMSNorm in a normalized readout, the identity in a raw one) and then the output projection W_{out}.

When r is a normalizer such as RMSNorm, it removes the scale of H_k (its magnitude), so $\mathcal{L}_k$ does not see the scale and gives it no gradient. The next loop, however, takes H_k itself as input, scale included.

Formally, write $H = su$, where $s = \|H\|_{\text{rms}}$ is the scale (its RMS norm) and $\|u\|_{\text{rms}} = 1$ is the direction. An unnormalized readout preserves scale ($r_{\text{raw}}(su) = su$), while a normalized readout removes it ($r_{\text{norm}}(su) = u$, up to the numerical epsilon).

Definition 1 (Radial-gradient fraction)**.** For a token with hidden state $H_k \neq 0$ at loop k, the radial-gradient fraction is

$$\rho_k = \frac{|\langle \nabla_{H_k} \mathcal{L}_k, H_k \rangle|}{\|\nabla_{H_k} \mathcal{L}_k\|_2 \, \|H_k\|_2} \in [0, 1],$$

the absolute cosine between the gradient of the local loss and the radial direction. It is zero when $\mathcal{L}_k$ gives no gradient on scale. We report its mean over tokens.

Lemma 1 (Blind spot: a scale-invariant readout gives no gradient on scale; Kanavalau et al., 2026, Proposition 4.1)**.** If $r(\alpha H) = r(H)$ for all $\alpha > 0$ and $\mathcal{L}(H) = \text{CE}(W_{\text{out}} r(H), y)$ is differentiable at $H \neq 0$, then

$$\langle \nabla_H \mathcal{L}(H), H \rangle = 0. \tag{2}$$

The proof is the chain rule applied to $\mathcal{L}(e^t H)$, which does not change with the scale t. With the small epsilon in real LayerNorm/RMSNorm the identity is only approximate, but the gradient on scale goes to zero quickly as the hidden norm grows.

Lemma 2 (Accumulation: pre-norm loops keep adding to scale)**.** Let $H = su$ with $\|u\|_{\text{rms}} = 1$. For the standard pre-norm residual update, where Norm is the pre-sublayer normalizer and B is the residual branch:

$$F(H) = H + B(\text{Norm}(H)), \qquad b(u) = B(\text{Norm}(u)),$$

we have $F(su) = su + b(u)$. Decomposing the residual update into scale (radial) and directional (angular) parts:

$$b(u) = a_{\mathrm{rad}}(u)u + b_{\perp}(u), \qquad a_{\mathrm{rad}}(u) = \frac{\langle u, b(u)\rangle}{d}, \qquad \langle u, b_{\perp}(u)\rangle = 0.$$

If $\|b(u)\|_{\mathrm{rms}} = O(1)$ as $s \to \infty$, then

$$\textit{scale update:} \qquad \|F(su)\|_{\mathrm{rms}} = s + a_{\mathrm{rad}}(u) + O(s^{-1}),$$
$$\textit{direction update:} \qquad \frac{F(su)}{\|F(su)\|_{\mathrm{rms}}} = u + \frac{b_{\perp}(u)}{s} + O(s^{-2}).$$

If the network learns a positive radial update ($a_{\mathrm{rad}}(u_k) > 0$), nothing in the local loss pushes back on the drift:

$$\langle \nabla_{H_k} \mathcal{L}_k, H_k\rangle = O(\epsilon/s_k^2), \qquad s_{k+1} - s_k = a_{\mathrm{rad}}(u_k) + O(s_k^{-1}). \tag{3}$$

Proof details are in Appendix A. This holds token by token, since RMSNorm normalizes each token separately (Appendix E).

Gradients from later loops. Losses at later loops do reach s_k through the recurrence, so Lemma 1 alone does not settle whether the full objective controls scale. Proposition 3 bounds this indirect signal for the single-token map of Lemma 2.

Proposition 3 (Later-loop gradients on scale are attenuated)**.** Let $H_{j+1} = F(H_j)$ with F as in Lemma 2, and write $H_j = s_j u_j$. Assume that (i) each later readout is exactly scale invariant, so $\mathcal{L}_j = \ell_j(u_j)$ for $j > k$; (ii) $\|b(u)\|_{\mathrm{rms}} \le B$ and b is L_b-Lipschitz in the RMS norm on $\{\|u\|_{\mathrm{rms}} = 1\}$; (iii) $|D\ell_j(u)[v]| \le G\|v\|_{\mathrm{rms}}$ for every u and tangent direction v; and (iv) the scale does not decrease, $s_{j+1} \ge s_j$ for $j \ge k$. Then, for the derivative that rescales H_k with u_k fixed and every $t \ge 1$,

$$\left|\frac{\partial \mathcal{L}_{k+t}}{\partial \log s_k}\right| \le \frac{G\,B\,t}{s_k} \exp\Big(\frac{t\,(L_b + B\,t)}{s_k}\Big), \tag{4}$$

which is at most $e\,G\,B\,t/s_k$ when $s_k \ge t\,(L_b + B\,t)$.

Once the scale is large, the whole remaining objective, and not only the local loss, sends a gradient on scale that falls off like $1/s_k$. Assumption (iv) describes the drift regime that we measure in Sections 4–6. The proof is in Appendix A.

Controlling scale. We can expose scale to a loss, with an unnormalized readout or an explicit norm penalty, or take it out of the recurrent path by normalizing the state between loops. Table 16 in Appendix H lists these options, and Sections 4–5 test them.

3 Experimental Setup

Our main probe experiments (44M and 129M parameters) train looped language models on WikiText-103 (Merity et al., 2017), each applying a shared decoder stack $K = 4$ times. Each recurrent step applies the same full stack: 8 pre-norm RMSNorm/SwiGLU transformer layers for the 44M model ($d = 512$, 8 heads), or 12 layers for the 129M model ($d = 768$, 12 heads). We denote the stack output after loop k by H_k. The output projection W_{out} (LM head) is shared across exits. A normalized readout computes logits from $\mathrm{RMSNorm}(H_k)$, and a raw readout computes them directly from H_k. There is no normalization between loops, so H_k goes into the next loop unnormalized and its scale stays in the residual stream. We train both sizes for 4 epochs over 3 seeds.

The main experiment crosses two choices: which loops receive the loss (only the final loop, *final-loop CE*, or every loop with the K cross-entropies averaged, *per-loop CE*) and whether the readout is normalized (RMSNorm) or raw. A raw readout changes two things at once: it lets the loss see scale, and it removes the normalizer from the readout. Two further models separate these. The *norm penalty* model keeps every readout normalized but adds the auxiliary loss $\lambda \frac{1}{K}\sum_{k=1}^{K} \mathbb{E}\,\|H_k\|_{\mathrm{rms}}^2$ with $\lambda = 0.01$, so the loss sees scale while the readout is unchanged. The *final-only norm* model keeps the final readout normalized, as in a standard language model, and uses raw readouts only

Table 1: **Dense CE does not control scale when every LM head is normalized.** WikiText-103 validation PPL and final-loop hidden-state norm at training depth $K = 4$. Values are mean $\pm$ std over 3 seeds. "Final-Loop" applies loss only to the last exit; "Per-loop" averages loss across all supervised exits. Compare quality within the same loss-placement group.

		44M		129M	
Loss	**LM Head**	$\|H_K\|_2$	PPL	$\|H_K\|_2$	PPL
Final-Loop	Normalized (RMSNorm)	5,674±393	5.40	5,997±25	4.88
Final-Loop	Unnormalized (Raw)	15±2	5.30	24±3	4.92
Per-loop	**Normalized (RMSNorm)**	**39,207±19,891**	6.04	**56,051±23,993**	5.35
Per-loop	Unnormalized (Raw)	44±1	5.38	77±4	4.92
Per-loop	Final-only norm	57±2	5.42	83±10	4.93
Final-Loop + penalty	Normalized (RMSNorm)	21	5.28	27±0	4.81
Per-loop + penalty	Normalized (RMSNorm)	17±0	5.44	22±0.3	4.91

at loops $1, \ldots, K-1$, so the loss sees scale at every loop except the last. We report cross-entropy and perplexity (PPL). Unless stated otherwise, norms are the Euclidean norm $\|H\|_2$ of each token's hidden state, averaged over tokens. Each table uses one evaluation setup throughout, and Appendix D gives the details.

Controlled 1.4B Study. To check whether all this holds at a larger size, we train four matched 1.435B-parameter models from scratch on FineWeb-Edu (Penedo et al., 2024). They follow the Ouro architecture (Zhu et al., 2025), with 24 shared layers, width 2048, and 16 attention heads. We remove Ouro's post-attention and post-MLP normalizers so the blocks are plain pre-norm residual blocks, and we put cross-entropy on all four loops. The four models share initialization, data order, batch size, and learning rate, and differ only in how they control scale (Section 5): the RMS head (normalized readout, the baseline), the raw head, the norm penalty, and inter-loop norm, which normalizes the state between loops. We evaluate the matched 6B-token checkpoints on 128 validation sequences.

4 Per-Loop Cross-Entropy Fails to Control Hidden State Norms

We cross loss placement with readout normalization in the 44M and 129M models (Table 1). If norm growth came only from supervising the last loop, per-loop CE would stop it. If the readout blind spot causes it, per-loop CE will not stop it while every readout is normalized, but exposing scale to the loss will.

With per-loop CE and a normalized readout, the final-loop norms reach 39,207 (44M) and 56,051 (129M), although every loop has its own loss. Removing the RMSNorm from the readout brings the norms back into the tens under both supervision modes.

Removing the normalizer is not the only option. Keeping the normalized readout and penalizing scale bounds the norms at 17–27, and the final-only norm model, which exposes scale at every loop except the last, keeps them at 57–83.

In Figure 1, per-loop CE makes the early exits usable but leaves the norm large, while exposing scale bounds the norm but, with final-loop CE alone, leaves the exits unusable. With per-loop CE and normalized readouts the token norms develop heavy tails, while unnormalized readouts keep them compact (Appendix E).

5 The Blind Spot at 1.4B Scale

We train four matched 1.4B models with per-loop cross-entropy (Section 3) that differ only in how they handle scale. The *RMS head* is the baseline with a normalized readout, the *raw head* removes the normalizer, the *norm penalty* keeps the normalized readout and adds the penalty with $\lambda = 0.01$, and *inter-loop norm* keeps the normalized readout and also normalizes H_k before the next loop. As in the released Ouro implementation (Zhu et al., 2025), one RMSNorm module serves both roles in this model.

Table 2: **The blind spot at 1.4B.** $\|H_4\|_2$ is the mean pre-readout L2 norm at loop 4 on 128 validation sequences (for inter-loop norm, measured before the boundary normalization), and r_4 is the local CE radial-gradient fraction at loop 4 on eight sequences. Matched 6B-token checkpoints, one training run per model.

Variant	$\|H_4\|_2$	r_4
RMS head	1,884	5.5×10^{-9}
Raw head	98	5.0×10^{-2}
Norm penalty	27	3.6×10^{-7}
Inter-loop norm	649	5.6×10^{-9}

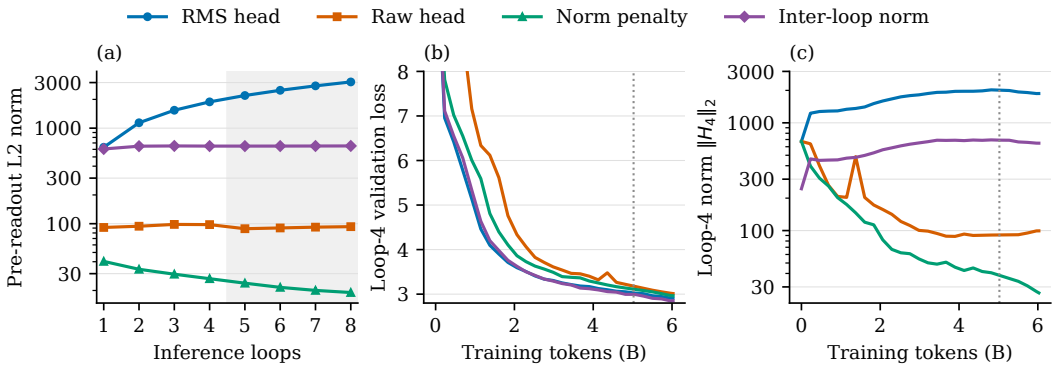

Figure 3: **Hidden-state norm and training curves at 1.4B.** (a) Mean tokenwise pre-readout L2 norm at each inference loop for the 6B-token checkpoints on 128 validation sequences. Shaded areas mark extrapolation beyond the training depth ($K = 4$). (b) Loop-4 validation loss and (c) loop-4 norm during training, recorded on 32 validation sequences. Early loss values above the axis are clipped; the dotted line indicates the final learning-rate decay at 5B tokens. Data reflects one training run per model variant.

Gradient fraction and norm. The RMS head and inter-loop norm pass almost no gradient to scale, with ρ_4 (Definition 1) near 10^{-9}, while the raw head has $\rho_4 = 5.0 \times 10^{-2}$ (Table 2). The norm penalty is at 3.6×10^{-7} because its hidden state is small, which makes the normalizer's epsilon matter more (Section 2). The RMS head's norm grows at every loop, from 631 at loop 1 to 3,046 at loop 8 (Figure 3a), and during training its loop-4 norm rises from 672 at initialization to about 2,000 by 4B tokens (Figure 3c). After the first billion tokens the other three stay below 700 and end at 99, 26, and 647.

Recipe transfer. To separate model size from the training recipe, we train the four variants with the 129M model's transformer stack under the 1.4B recipe, with two seeds and two weight decays (Table 3; details in Appendix C). The mechanism matches: at loop 4, ρ_4 is 8.6×10^{-9} for the RMS head and 5.6×10^{-2} for the raw head, and only the RMS head's norm grows across loops. Under the small-model recipe the raw head is 8% better than the RMS head at 129M (Table 1), but under the 1.4B recipe it is 2.9–5.3% worse in all four runs, and the norm penalty stays within 1% of the RMS head. Lowering weight decay from 0.1 to 0.01 does not change this (Table 4).

6 Mechanism Diagnostics

Radial-gradient fraction. With a normalized readout the radial-gradient fraction ρ_k (Definition 1) is about 10^{-8}, and with an unnormalized readout about 10^{-2}. The 1.4B models show the same pattern, and their scale sweep leaves the loss unchanged under a normalized readout (Section 5 and Table 7 in Appendix B).

Table 3: Recipe transfer to approximately 189M total parameters: loop-4 PPL on the same 128-sequence harness, after 537,886,720 training tokens per run. Relative differences use the RMS-head baseline with the same seed.

Variant	Seed 42	Δ	Seed 43	Δ
RMS head	51.016	–	51.330	–
Raw head	52.952	+3.79%	52.826	+2.91%
Norm penalty	50.751	−0.52%	50.815	−1.00%
Inter-loop norm	51.939	+1.81%	51.344	+0.03%

Table 4: Weight-decay follow-up to Table 3: the same runs with AdamW weight decay 0.01 instead of 0.1, loop-4 PPL on the same 128-sequence harness. The last column is the seed-mean loop-4 pre-readout L2 norm at weight decay 0.1 and 0.01.

Variant	Seed 42	Δ	Seed 43	Δ	Norm
RMS head	51.210	–	51.794	–	303 → 482
Raw head	53.916	+5.28%	53.991	+4.24%	133 → 126
Norm penalty	51.154	−0.11%	52.229	+0.84%	27 → 27
Inter-loop norm	52.511	+2.54%	52.259	+0.90%	115 → 167

Radial update. We split each loop's residual update into a radial part, which changes the scale, and an angular part, which changes the direction. In the per-loop checkpoints with normalized readouts the radial part is positive at every loop, as Lemma 2 predicts. At 44M it predicts that the RMS scale $s = \|H\|_{\mathrm{rms}}$ grows by about 258 per loop, within 3% of the measured growth, while the direction barely moves. Appendix G gives the 30-loop trajectories and the decomposition.

Clamping the scale. After loop 1 we rescale each token's hidden state back to its loop-1 RMS norm before both the readout and the next loop. This removes the scale added after loop 1 and keeps every change in direction.

With a normalized readout the cross-entropy changes by at most 0.0005 (Table 5), so the scale that builds up carries almost no predictive information at evaluation time. The clamp tests only the scale added after loop 1, not every radial part of the update.

7 Exit Training and Depth Extrapolation

With cross-entropy only on the final loop, the intermediate predictions are unusable, even with unnormalized readouts (Figure 1 and Appendix J). Per-loop supervision makes the early exits usable at every size we tested: at 1.4B, every model's loop-2 PPL is within 0.6% of its loop-4 PPL.

Under the small-model recipe, models with normalized readouts barely improve from loop 1 to loop 4, while the three scale-control variants improve steadily (Table 6; Figure 6 in Appendix K). Under the 1.4B recipe the ranking changes. At 1.4B the drifting baseline gains 3.3%, the raw head 2.7%, the penalty nothing, and inter-loop norm 10.0% (Table 8 in Appendix B). In the recipe-transfer runs, inter-loop norm gains the most from its loops but is not better than the baseline at loop 4 (Appendix C). We therefore report early-exit usability, scale control, and the gain from extra loops as separate properties.

Our fixed-depth evaluations use teacher-forced scoring, and the threshold sweep in Appendix I is an in-sample calibration check, it does not measure KV-cached generation speed.

8 Related Work

Looped transformers. Universal Transformers (Dehghani et al., 2019) combined weight sharing across depth with adaptive computation, and Huginn (Geiping et al., 2025) and Ouro (Zhu et al., 2025) scale the idea to recurrent-depth language models. In its released implementation, Ouro also normalizes the state between loops, like our inter-loop norm model. Looping has been studied for

Table 5: **Clamping recurrent scale has almost zero effect on cross-entropy.** Evaluated on per-loop CE models. The norm ratio is the clamped $K = 4$ norm divided by the unclamped $K = 4$ norm; ΔCE is the clamped minus unclamped cross-entropy. Values are means over 3 seeds.

Readout	**Model**	$K = 4$ **norm ratio**	Δ**CE**
Normalized	44M	0.40	$+0.0004$
Normalized	129M	0.55	$+0.0005$
Unnormalized	44M	0.53	$+0.0055$
Unnormalized	129M	0.46	$+0.0015$
Final-only norm	44M	0.41	$+0.0035$
Final-only norm	129M	0.42	$+0.0027$

Table 6: **Scale-controlled per-loop models improve with depth and stay lower-PPL.** Variable-depth inference on per-loop-loss models, mean PPL over 3 seeds. Both depths use the same fixed-depth evaluation slice; no calibrated halting threshold is involved. Terminal-only variable-depth baselines are in Appendix J.

Readout	**Model**	$K = 1$	$K = 4$	ΔPPL
RMSNorm	44M	5.58	5.59	$+0.01$
RMSNorm	129M	4.98	4.97	-0.01
Raw	44M	5.14	4.94	-0.20
Raw	129M	4.68	4.55	-0.13
Final-only norm	44M	5.20	5.00	-0.20
Final-only norm	129M	4.70	4.55	-0.15
Norm penalty	44M	5.22	5.00	-0.22
Norm penalty	129M	4.67	4.53	-0.14

reasoning (Saunshi et al., 2025; Kohli et al., 2026), with a depth budget (Jeddi et al., 2026), with intra-loop self-distillation for visual generation (Goyal et al., 2026), and in large mixture-of-experts models (Gao et al., 2026).

Exits and stopping. ACT (Graves, 2016), PonderNet (Banino et al., 2021), and depth-adaptive Transformers (Elbayad et al., 2020) learn how much computation to spend, and more recent work stops when the output distribution converges (Logan, 2026), uses randomized loop counts and learned stopping on algorithmic tasks (Kuo et al., 2026), or assigns reinforcement-learning credit to intermediate states (Williams & Tureci, 2026; Tang et al., 2026). Popescu et al. (2026) separate forming the recurrent trajectory from choosing when to exit, and find that learned halting weights can hurt trajectory training. Their readout decides when to exit, while ours is the LM head, which decides what the cross-entropy can see.

Residual scale and normalization. Normalization placement (Xiong et al., 2020), NormFormer (Shleifer et al., 2021), DeepNet (Wang et al., 2022), and Peri-LN (Kim et al., 2025) change where normalization goes or how residuals are scaled to make optimization easier. The Curse of Depth (Sun et al., 2025) links growing activation variance in pre-norm models to later layers contributing less, nGPT (Loshchilov et al., 2024) keeps representations on a hypersphere, and Norm-Agnostic Residual Networks (NAG; Figliolia & Millidge, 2026) separate magnitude from direction because residual-norm growth shrinks the relative size of each update. RMSNorm (Zhang & Sennrich, 2019) is invariant to input rescaling up to its numerical epsilon, and TaperNorm (Kanavalau et al., 2026) shows that final normalization removes radial loss gradients, while removing it allows cross-entropy to decrease by increasing logit magnitudes. In a loop the pre-readout state is reused, so that same final normalization hides recurrent magnitude from each exit's local cross-entropy, even with a loss at every loop.

Stability of looped dynamics. Analyses of recurrent gradients (Pascanu et al., 2013) and of looped models (Blayney et al., 2026; Labovich, 2026) study these dynamics directly. Fully Looped Transformer (Fu et al., 2026), Parcae (Prairie et al., 2026), and STARS (Yang et al., 2026) target residual explosion, injection spectra, or Jacobian stability, and DeepLoop (Li et al., 2026) derives

residual scaling and initialization for gradients summed over repeated visits to the same parameters. Fractal basins (Lai et al., 2026) point to transient dynamics near almost-correct solutions as another source of long latent reasoning.

9 LIMITATIONS

The 1.4B study is one training run per model, stopped at 6B tokens, with a learning rate and a penalty weight that we did not tune for each model, so we only use it for the norm and gradient measurements. Our gradient measurements only look at each loop's own loss, so we have not measured how much gradient the later losses send back to the scale through the recurrence. We also do not know which part of the training recipe decides whether scale control improves perplexity. The recipe-transfer runs rule out model size and weight decay, but not the other differences. Finally, every model is trained with four loops and we only test more loops at inference, so we do not know how scale drifts when training itself uses more loops.

10 CONCLUSION

Applying cross-entropy at every iteration of a looped language model is effective for training intermediate predictions, but it does not control the scale of the recurrent state. Because standard readouts normalize the hidden state before evaluating the loss, they mask the recurrent scale from the optimization signal. In every baseline configuration we tested, this readout blind spot allowed the hidden state norm to accumulate at every loop.

We demonstrate that controlling this scale, either by making it visible to the loss or by resetting it internally, resolves the norm accumulation. However, bounding the norm does not guarantee a stronger language model; the downstream effect of scale control is highly sensitive to the training recipe. Future work should complete the 1.4B evaluations with multiple seeds and isolate exactly which recipe components determine when scale-control interventions improve absolute perplexity.

AI USE STATEMENT

We used generative AI tools (an AI coding assistant) to write and debug parts of the training and evaluation code, and to prepare and monitor cluster training jobs. We also used them to generate tables and figures from saved evaluation outputs, to search the literature, and to edit the text. We also used generative AI tools to check our mathematical derivations. We reviewed all AI-assisted work: code was checked with unit tests and against independent evaluations, and every reported number is regenerated from archived evaluation outputs. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies training dynamics of language models on public text corpora (WikiText-103 and FineWeb-Edu). It involves no human subjects and releases no new data, and we are not aware of ethical concerns specific to it.

REPRODUCIBILITY STATEMENT

Section 3 describes the models, data, and training setups. Appendix A gives full proofs of the lemmas in Section 2. Appendix D lists the evaluation harnesses and slices, Appendix B gives the 1.4B checkpoints, evaluation protocol, and downstream scoring, and Appendix C gives the recipe-transfer and weight-decay runs. All tables and the 1.4B figure are generated by a script from archived evaluation outputs.

REFERENCES

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

Hugh Blayney, Álvaro Arroyo, Johan Obando-Ceron, et al. A mechanistic analysis of looped reasoning language models. *arXiv preprint arXiv:2604.11791*, 2026.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020.

Tomás Figliolia and Beren Millidge. Scaling adaptive depth with norm-agnostic residual networks. *arXiv preprint arXiv:2606.16112*, 2026. URL https://arxiv.org/abs/2606.16112.

Rao Fu, Zixuan Yang, Jiankun Zhang, Jing Ma, Hechang Chen, Yu Li, and Yi Chang. Simply stabilizing the loop via fully looped transformer. *arXiv preprint arXiv:2605.18797*, 2026.

Zitian Gao, Yilong Chen, Yihao Xiao, Xinyu Yang, Ran Tao, Joey Zhou, and Bryan Dai. Loop the loopies! *arXiv preprint arXiv:2607.16051*, 2026. URL https://arxiv.org/abs/2607.16051.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Sahil Goyal, Swayam Agrawal, Gautham Govind Anil, Prateek Jain, Sujoy Paul, and Aditya Kusupati. ELT: Elastic looped transformers for visual generation. *arXiv preprint arXiv:2604.09168*, 2026.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

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

Andrei Kanavalau, Carmen Amo Alonso, and Sanjay Lall. Gated normalization removal and scale anchoring in pre-norm transformers. *arXiv preprint arXiv:2602.10408*, 2026.

Jeonghoon Kim, Byeongchan Lee, Cheonbok Park, Yeontaek Oh, Beomjun Kim, Taehwan Yoo, Seongjin Shin, Dongyoon Han, Jinwoo Shin, and Kang Min Yoo. Peri-LN: Revisiting normalization layer in the transformer architecture. *arXiv preprint arXiv:2502.02732*, 2025. URL https://arxiv.org/abs/2502.02732.

Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers. *arXiv preprint arXiv:2604.07822*, 2026.

Hsun-Yu Kuo, El Mahdi Chayti, Patrik Reizinger, Wieland Brendel, and Martin Jaggi. Stabilizing extrapolation in looped transformers via learned stochastic stopping. *arXiv preprint arXiv:2606.29983*, 2026. URL https://arxiv.org/abs/2606.29983.

Asher Labovich. Stability and generalization in looped transformers. *arXiv preprint arXiv:2604.15259*, 2026.

Jeffrey Lai, Anthony Bao, John Quinn, and William Gilpin. Fractal basins trap latent reasoning. *arXiv preprint arXiv:2609.04963*, 2026. URL https://arxiv.org/abs/2609.04963.

Shuzhen Li, Yifan Zhang, Jiacheng Guo, Quanquan Gu, and Mengdi Wang. DeepLoop: Depth scaling for looped transformers. *arXiv preprint arXiv:2607.13491*, 2026. URL https://arxiv.org/abs/2607.13491.

Joe Logan. Per-token fixed-point convergence in depth-recurrent transformers. *arXiv preprint arXiv:2607.14427*, 2026. URL https://arxiv.org/abs/2607.14427.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. *arXiv preprint arXiv:2410.01131*, 2024. URL https://arxiv.org/abs/2410.01131.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pp. 1310–1318, 2013.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024.

Andrei Cristian Popescu, Haitz Sáez de Ocáriz Borde, and Pietro Liò. Adaptive depth in looped transformers: Diagnosing learned halting gates and trajectory readouts. *arXiv preprint arXiv:2607.20519*, 2026. URL https://arxiv.org/abs/2607.20519.

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.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025.

Sam Shleifer, Jason Weston, and Myle Ott. NormFormer: Improved transformer pretraining with extra normalization. *arXiv preprint arXiv:2110.09456*, 2021.

Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. *arXiv preprint arXiv:2502.05795*, 2025. URL https://arxiv.org/abs/2502.05795.

Guo Tang, Shixin Jiang, Heng Chang, Nuo Chen, Yuhan Li, Huiming Fan, Jia Li, Ming Liu, and Bing Qin. LoopRPT: Reinforcement pre-training for looped language models. *arXiv preprint arXiv:2603.19714*, 2026.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. DeepNet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.

Jonathan Williams and Esin Tureci. Prioritize the process, not just the outcome: Rewarding latent thought trajectories improves reasoning in looped language models. *arXiv preprint arXiv:2602.10520*, 2026.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, 2020.

Xiao-Wen Yang, Ziyu Han, Xi-Hua Zhang, Wen-Da Wei, Jie-Jing Shao, Lan-Zhe Guo, and Yu-Feng Li. Stabilizing recurrent dynamics for test-time scalable latent reasoning in looped language models. *arXiv preprint arXiv:2605.26733*, 2026.

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

Rui-Jie Zhu, Zixuan Wang, Kai Hua, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

The appendices give reproducibility details, supporting ablations, and diagnostic checks for the main results.

A Proof Details for Section 2

Visibility lemma. Define $\phi(t) = \mathcal{L}(e^t H)$. If the readout is scale-invariant, $r(e^t H) = r(H)$ for all t, so $\phi(t)$ is constant. Therefore $\phi'(0) = 0$. By the chain rule,

$$\phi'(0) = \langle \nabla_H \mathcal{L}(H), H \rangle , \tag{5}$$

which proves Lemma 1. With RMSNorm/LayerNorm epsilon terms, scale invariance is approximate rather than exact; at large scale the epsilon contribution is suppressed, giving the $O(\epsilon/s^2)$ radial term used in the main text.

Activity lemma. For $H = su$ with $\|u\|_{\mathrm{rms}} = 1$ and $\|v\|_{\mathrm{rms}}^2 = d^{-1}\|v\|_2^2$, ignoring normalizer epsilons gives

$$F(su) = su + b(u). \tag{6}$$

Decompose the residual update into radial and orthogonal parts,

$$b(u) = a_{\mathrm{rad}}(u)u + b_\perp(u), \qquad a_{\mathrm{rad}}(u) = \frac{\langle u, b(u)\rangle}{d}, \qquad \langle u, b_\perp(u)\rangle = 0. \tag{7}$$

Then

$$\|su + b(u)\|_{\mathrm{rms}}^2 = s^2 + 2s\, a_{\mathrm{rad}}(u) + \|b(u)\|_{\mathrm{rms}}^2. \tag{8}$$

Taking the square root and using $\|b(u)\|_{\mathrm{rms}} = O(1)$ in s gives

$$\|F(su)\|_{\mathrm{rms}} = s + a_{\mathrm{rad}}(u) + O(s^{-1}). \tag{9}$$

Substituting this denominator into

$$\frac{F(su)}{\|F(su)\|_{\mathrm{rms}}} = \frac{(s + a_{\mathrm{rad}}(u))u + b_\perp(u)}{s + a_{\mathrm{rad}}(u) + O(s^{-1})} \tag{10}$$

gives

$$\frac{F(su)}{\|F(su)\|_{\mathrm{rms}}} = u + \frac{b_\perp(u)}{s} + O(s^{-2}), \tag{11}$$

which proves Lemma 2.

Later-loop gradients. We prove Proposition 3. Write $P_j = I - u_j u_j^\top/d$ for the projection orthogonal to u_j, so that $\|P_j x\|_{\mathrm{rms}} \le \|x\|_{\mathrm{rms}}$. For a perturbation ∂H_j of the state, differentiating $u_j = H_j/\|H_j\|_{\mathrm{rms}}$ and $\log s_j = \log \|H_j\|_{\mathrm{rms}}$ gives

$$\partial u_j = \frac{P_j\, \partial H_j}{s_j}, \qquad |\partial \log s_j| = \frac{|\langle u_j, \partial H_j\rangle|}{d\, s_j} \le \frac{\|\partial H_j\|_{\mathrm{rms}}}{s_j}. \tag{12}$$

Rescale H_k by e^δ with u_k fixed and differentiate at $\delta = 0$. Let $\sigma_j = |\partial \log s_j|$ and $\varepsilon_j = \|\partial u_j\|_{\mathrm{rms}}$, so that $\sigma_k = 1$ and $\varepsilon_k = 0$. Since $H_{j+1} = s_j u_j + b(u_j)$ by Lemma 2,

$$\partial H_{j+1} = (\partial \log s_j)\, s_j u_j + s_j\, \partial u_j + J_b(u_j)\, \partial u_j, \tag{13}$$

where J_b is the derivative of b along the unit-RMS sphere, and $\|J_b\| \le L_b$ by assumption (ii). Because H_{j+1} is parallel to u_{j+1}, we have $P_{j+1}H_{j+1} = 0$, and hence $P_{j+1}s_j u_j = -P_{j+1}b(u_j)$. Therefore

$$\partial u_{j+1} = \frac{-(\partial \log s_j)\, P_{j+1} b(u_j) + P_{j+1}\big(s_j I + J_b(u_j)\big)\partial u_j}{s_{j+1}}. \tag{14}$$

With $\|b\|_{\mathrm{rms}} \le B$ and $s_{j+1} \ge s_j \ge s_k$ from assumption (iv), this and the bound on $|\partial \log s_{j+1}|$ give

$$\varepsilon_{j+1} \le a\, \sigma_j + M\varepsilon_j, \qquad \sigma_{j+1} \le \sigma_j + M\varepsilon_j, \qquad a = \frac{B}{s_k}, \quad M = 1 + \frac{L_b}{s_k}. \tag{15}$$

Let $\lambda = M(1 + a\, t)$. We show by induction that $\sigma_{k+n} \le \lambda^n$ and $\varepsilon_{k+n} \le a\, n\, \lambda^n$ for $0 \le n \le t$. The case $n = 0$ holds. For the step, $\varepsilon_{k+n+1} \le a\lambda^n(1 + Mn) \le a(n+1)\lambda^{n+1}$, because $\lambda \ge M \ge 1$, and $\sigma_{k+n+1} \le \lambda^n(1 + Ma\, n) \le \lambda^{n+1}$, because $\lambda \ge 1 + Ma\, t \ge 1 + Ma\, n$. By assumption (i), $\mathcal{L}_{k+t} = \ell_{k+t}(u_{k+t})$, and by assumption (iii),

$$\left|\frac{\partial \mathcal{L}_{k+t}}{\partial \log s_k}\right| \le G\, \varepsilon_{k+t} \le G\, a\, t\, \lambda^t = \frac{G\, B\, t}{s_k}\Big(1 + \frac{L_b}{s_k}\Big)^t \Big(1 + \frac{B\, t}{s_k}\Big)^t \le \frac{G\, B\, t}{s_k} \exp\Big(\frac{t\,(L_b + B\, t)}{s_k}\Big). \tag{16}$$

When $s_k \ge t\,(L_b + B\, t)$ the exponent is at most one, which gives the bound $e\, G\, B\, t/s_k$.

B CONTROLLED 1.4B STUDY: ADDITIONAL RESULTS

Checkpoints and evaluation slices. Every main 1.4B result uses the matched checkpoint at step 6,540, or 6,000,476,160 training tokens, from each arm's common schedule. Training continued to step 6,566; the reported checkpoint is the saved 6B milestone rather than the last checkpoint. The PPL harness uses 128 sequences of 2048 next-token targets (262,144 targets) from WikiText-103 validation, tokenized with the Ouro tokenizer. The first eight sequences are used for local radial gradients and final-state scale sweeps. Their absolute CE need not match the larger PPL slice. Table 7 reports the scale sweep, and Table 8 gives next-token PPL at every inference depth.

Table 7: Final-loop CE under hidden-state rescaling on eight validation sequences. This diagnostic slice differs from the 128-sequence PPL evaluation. Values are unchanged to the saved precision for normalized heads.

Variant	$0.25\times$	$0.5\times$	$1\times$	$2\times$	$4\times$
RMS head	2.58509	2.58509	2.58509	2.58509	2.58509
Raw head	7.14177	4.25566	2.68710	3.71044	6.81065
Norm penalty	2.63780	2.63780	2.63782	2.63782	2.63782
Inter-loop norm	2.52995	2.52995	2.52995	2.52995	2.52995

Table 8: Next-token perplexity by inference loop for the matched 6B-token checkpoints; the training depth is four. All values use the same 128-sequence evaluation.

Variant	1	2	3	4	5	6	7	8
RMS head	20.999	20.189	20.155	20.297	20.523	20.811	21.153	21.545
Raw head	23.214	22.686	22.577	22.585	22.914	23.285	23.863	24.635
Norm penalty	21.595	21.477	21.494	21.614	22.131	23.457	25.632	28.536
Inter-loop norm	21.204	19.150	18.911	19.093	19.538	20.198	21.032	22.016

Local gradients and carried-state norms. For each token we compute $|\langle \nabla_h \mathcal{L}_k, h\rangle|/(\|\nabla_h \mathcal{L}_k\|_2 \|h\|_2)$ through that loop's output head, then average over tokens. The hidden state is detached from the recurrent graph for this diagnostic; it does not measure gradients from later losses. The penalty arm's reported gradient is CE-only. For inter-loop normalization, all reported norms refer to the stack output before either output or boundary normalization; they are not the magnitude entering the next loop. The implementation's update ratio compares each output with its carried input, which is normalized only in the inter-loop arm. We therefore do not use that ratio for a cross-arm comparison of directional progress.

Downstream scoring. Tables 9 and 10 report a custom zero-shot continuation scorer, implemented in `src/eval/loop1p4b_eval.py` with task loaders and prompts from `src/eval/ouro_downstream_eval.py`. For each candidate, only continuation tokens contribute to the sum of next-token log probabilities; the length-normalized variant divides by the number of scored tokens. Inputs are limited to 2048 tokens, and task loaders select validation splits. PIQA uses the `gimmaru/piqa` mirror with 1,000 evaluated examples; the actual counts for every task are provided rather than describing all tasks as complete canonical validation sets. The same examples are used at loops 2, 3, and 4. Macro accuracy averages the nine task accuracies equally. The small differences between arms are descriptive; these results do not establish statistical significance or constitute an evaluation with the standard LM Evaluation Harness.

Published-checkpoint diagnostic. The final-readout scale sweep on the published Ouro checkpoint is a separate diagnostic. Figure 4 illustrates its normalized head's scale invariance; the controlled experiment above provides the retraining comparison.

Table 9: Nine-task macro accuracy (percent), with summed continuation log-likelihood (Acc.) or token-length normalization (Norm.). Same tasks and examples at each loop; custom scorer, one training seed. These are descriptive averages, not significance tests.

Variant	Acc. 2	Acc. 3	Acc. 4	Norm. 2	Norm. 3	Norm. 4
RMS head	45.76	45.85	45.69	46.42	46.55	46.46
Raw head	45.12	45.04	44.91	45.89	46.02	45.93
Norm penalty	44.87	44.84	44.87	45.94	45.92	45.94
Inter-loop norm	45.76	45.95	45.94	46.79	47.16	47.22

Table 10: Loop-4 length-normalized accuracy (percent) and actual example counts. PIQA uses 1,000 evaluated examples; task splits and prompts follow the custom scorer described in the text.

Task	n	RMS	Raw	Penalty	Inter-loop
PIQA	1,000	65.40	64.50	65.20	66.20
ARC-Easy	570	51.75	53.51	52.98	53.33
ARC-Challenge	299	29.10	28.09	30.77	29.77
HellaSwag	10,042	36.63	34.40	35.49	38.58
OpenBookQA	500	30.00	29.20	28.80	30.40
BoolQ	3,270	61.25	61.56	59.97	61.71
WinoGrande	1,267	51.70	52.41	50.36	52.01
CommonsenseQA	1,221	31.53	30.79	30.88	30.55
SciQ	1,000	60.80	58.90	59.00	62.40

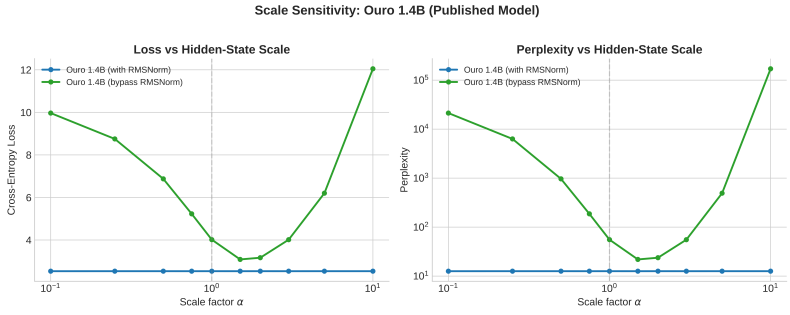

Figure 4: Final-hidden-state rescaling on the published Ouro 1.4B checkpoint. The normalized head is insensitive to scale, while removing its output normalization makes CE scale-sensitive. This is an output-head diagnostic, not a retraining comparison.

C Recipe Transfer at Smaller Scale

The old 129M study and the new 1.4B study differ in training data, tokenizer, sequence length, optimizer settings, positional encoding, dropout, embedding tying, and training budget. To test whether raw's quality disadvantage requires billion-parameter size, we train the four new arms with the smaller 12-layer, width-768 decoder stack under the new recipe, using seeds 42 and 43. The vocabulary has 49,152 entries and embeddings are untied, making the total parameter count approximately 189M; this is not a parameter-matched rerun of the old 129M model. Each run sees 537,886,720 tokens over 6,566 steps, with 40 sequences of length 2048 per step. The schedule uses the same 2,000-step warmup and an 89,292,800-token cooldown starting at step 5,476. Table 3 uses the new 128-sequence evaluation harness.

Raw has higher PPL in both seeds, so that disadvantage does not require billion-parameter size. The penalty slightly improves PPL, while inter-loop normalization ties or worsens it. The combined preregistered recipe-driven criterion required both raw and penalty to worsen PPL by more than 2% in both seeds; the opposite criterion required both to improve by more than 2%. Neither is met, so the combined outcome is mixed. The experiment does not identify a single causal recipe factor. Inter-loop normalization gains the most from its loops (5.87% and 5.78% lower PPL at loop 4 than at loop 1, against 1.54% and 0.73% for the RMS head), yet its loop-4 PPL is no better than the RMS head's in either seed. A large improvement relative to an arm's own first exit is therefore distinct from better absolute quality.

The mechanism measurements transfer to this size. At loop 4, the radial fraction is 8.6×10^{-9} for the RMS head, 1.2×10^{-8} for inter-loop normalization, 1.5×10^{-7} for the penalty, and 5.6×10^{-2} for the raw head. The RMS head's mean norm grows at every loop (from 115 at loop 1 to 306 at loop 4 and 518 at loop 8, seed 42), while the raw head stays near 134, the penalty shrinks from 28 to 22, and inter-loop normalization is flat after loop 1.

Weight-decay follow-up. Eight further runs repeat Table 3 with AdamW weight decay 0.01, the value used in the small-model study, leaving all other settings unchanged (Table 4). The preregistered weight-decay criterion required raw to improve PPL by more than 2% in both seeds; instead its disadvantage widens to 5.28% and 4.24%. Lowering weight decay raises the RMS head's loop-4 norm only from 303 to 482, far below the tens of thousands in Table 1. Weight decay therefore explains neither raw's advantage in the small-model recipe nor that recipe's much larger drift. The remaining differences between the recipes (data and tokenizer, sequence length, dropout, embedding tying, positional encoding, and training budget) are not isolated here.

D Reproducibility Details

Table 11: Implementation and evaluation details. The 1.4B arms use matched token order and training steps; their saved 6B milestone is the common evaluation checkpoint.

Component	Setting
44M / 129M data	WikiText-103 raw train/validation; regex tokenizer fit from text with maximum vocabulary 20k and minimum token frequency 2; sequence length 256 and stride 128.
44M / 129M training	$K = 4$ loops; AdamW with learning rate 3×10^{-4}, weight decay 0.01, gradient clipping 1.0, dropout 0.1; 4 epochs; 3 seeds; bfloat16 (bf16) on CUDA for full runs. Batch size is 64 for 44M and 32 for 129M.
44M / 129M evaluation	Table 1 reports logged validation at $K = 4$: final-exit CE for terminal supervision and mean per-loop CE for dense supervision, exponentiated to PPL and averaged over seeds. Table 6 uses the variable-depth evaluation harness on selected 44M and 129M checkpoints; comparisons within each row use the same checkpoint and evaluation code.
Radial clamp	Table 5 uses the same per-loop-loss checkpoints as Table 6. We run 20 batches of 8 sequences, 40,960 tokens per checkpoint, run loop 1 normally, and then clamp later tokenwise RMS scales to their loop-1 values before decoding and recurrence.
Dynamic halting	Sequence-level halting uses logit margin as the confidence score. The recovered threshold sweep reuses its calibration slice for evaluation; Appendix I reports it as an in-sample sensitivity check.
1.4B architecture	24 layers, width 2048, 16 heads, SwiGLU, RoPE, vocabulary 49,152; random initialization; plain pre-norm blocks; four loops with equally weighted next-token CE. Inter-loop normalization is enabled only in its designated arm.
1.4B training	FineWeb-Edu, Ouro tokenizer, sequence length 2048, global batch 448 sequences (917,504 tokens); AdamW, $\beta = (0.9, 0.95)$, weight decay 0.1, gradient clip 1.0, peak LR 3×10^{-4}, 2,000-step warmup; fp32 weights and bf16 autocast. One common seed and identical data order.
1.4B schedule	Initial cosine schedule planned for 14B tokens; after approximately 5B tokens (step 5,476), linear cooldown to 3×10^{-5} over approximately 1B additional tokens, ending at step 6,566.
1.4B checkpoint	All reported arms: step 6,540, 6,000,476,160 tokens. The saved 6B milestone precedes the end of the common cooldown by 26 steps.
1.4B evaluation	WikiText-103 validation, 128 sequences for PPL/norms; eight for local radial gradients and scale sweeps; inference depths 1–8. Nine-task custom continuation scoring with actual sample counts in Table 10.

E TOKEN-LEVEL NORM STATISTICS

The mean $\|H_K\|$ in Table 1 aggregates over tokens and can mask per-token outliers, which is a real concern given that real LayerNorm and RMSNorm operate per token. Table 12 reports the per-token $\|H_K\|_2$ distribution on $20{,}480$ validation tokens for each 2×2 condition (single seed; the across-seed pattern is the same).

Table 12: **Per-loop RMSNorm drift is a token-level heavy-tail failure, not just a mean shift.** Per-token final-loop norm distribution (Euclidean $\|H_K\|_2$) on 20,480 WikiText-103 validation tokens. RMSNorm-readout drift is concentrated in a heavy tail of outlier tokens, especially under per-loop CE: the mean is approximately 12–21$\times$ the median, with 1% of tokens reaching 10^5+. Raw readouts produce tighter bounded distributions in every condition.

Model	Condition	Mean	Median	p99	Max	Std
44M	Terminal+RMSNorm	5,644	5,493	8,768	11,798	899
44M	Terminal+Raw	**12**	**6**	**44**	**259**	**12**
44M	Per-loop+RMSNorm	63,347	2,986	342,685	375,295	112,383
44M	Per-loop+Raw	**42**	**8**	**150**	**390**	**46**
129M	Terminal+RMSNorm	5,888	5,805	10,636	16,232	1,796
129M	Terminal+Raw	**21**	**9**	**69**	**305**	**18**
129M	Per-loop+RMSNorm	26,426	2,132	124,507	149,790	43,792
129M	Per-loop+Raw	**71**	**10**	**259**	**808**	**82**

The two RMSNorm-readout failure modes are qualitatively different. Under terminal-only CE, RMSNorm drift is uniform: median $\approx$ mean and the p99 is within $2\times$ of the median. Under per-loop CE, RMSNorm drift is heavy-tailed: the medians are approximately 2,100–3,000 but the means are 12–21$\times$ larger and a small fraction of tokens reach 10^5+. Both raw conditions stay tight at every percentile.

F LOSS-PLACEMENT/SCALE DECOMPOSITION METRICS

Table 13 gives the exact values behind Figure 1. The CE gap is $\mathrm{CE}(K{=}1) - \mathrm{CE}(K{=}4)$, so smaller is better for early-exit usability.

Table 13: Exact loss-placement/scale decomposition values for the controlled 2×2, evaluated with the variable-depth slice harness (same as Table 6); absolute PPL uses a different slice and exit aggregation from the training-time full-validation values in Table 1. Per-loop CE marks whether cross-entropy is applied at every loop. Scale-visible marks raw readouts, which expose hidden-state magnitude to the CE loss.

Model	Condition	Per-loop CE	Scale-visible	K=1 PPL	K=4 PPL	CE gap	r_K
44M	Terminal + RMSNorm	–	–	1.1×10^9	5.04	19.21	4.1×10^{-9}
44M	Terminal + raw	–	✓	6.9×10^{16}	4.97	37.17	6.8×10^{-2}
44M	Per-loop + RMSNorm	✓	–	5.59	5.60	0.00	4.7×10^{-9}
44M	Per-loop + raw	✓	✓	5.16	4.97	0.04	7.2×10^{-2}
129M	Terminal + RMSNorm	–	–	226	4.55	3.91	6.9×10^{-9}
129M	Terminal + raw	–	✓	4.7×10^{12}	4.57	27.66	5.6×10^{-2}
129M	Per-loop + RMSNorm	✓	–	4.99	4.98	0.00	7.5×10^{-9}
129M	Per-loop + raw	✓	✓	4.71	4.59	0.02	4.7×10^{-2}

G RESIDUAL-UPDATE DECOMPOSITION

Table 14 reports the direct measurement of the Lemma 2 terms on trained checkpoints. The update magnitude is averaged over loops $k \geq 10$, after the initial transient. The table is not meant to make Lemma 2 look deep; it checks that the trained checkpoints enter the large-scale regime where the expansion is informative.

Table 14: **Normalized-readout checkpoints learn much larger radial residual updates.** Direct measurement of the residual-update decomposition terms from Lemma 2.

Model	Condition	a_{rad}	$\|b_\perp\|_{\text{rms}}$	$\|b\|_{\text{rms}}$	$\|b\|_{\text{rms}}/\|a_{\text{rad}}\|$
44M	Per-loop + Raw	0.27	0.05	0.28	1.01
44M	Per-loop + RMSNorm	252.65	15.45	253.12	1.00
44M	Terminal + Raw	−0.02	0.06	0.07	3.74
44M	Terminal + RMSNorm	210.81	5.35	210.88	1.00
129M	Per-loop + Raw	0.51	0.08	0.51	1.01
129M	Per-loop + RMSNorm	13.99	14.86	20.41	1.46

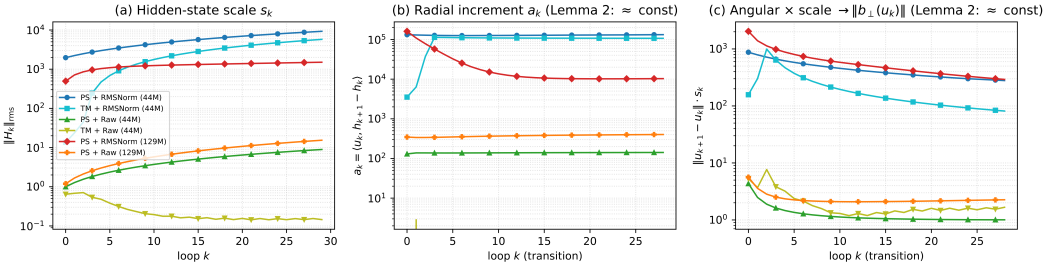

Figure 5: **Trained checkpoints enter the slow-angular-motion regime predicted by the scale expansion.** (a) Hidden-state RMS scale over 30 loops. (b) Euclidean radial residual inner product $\tilde{a}_k = \langle u_k, F(H_k) - H_k \rangle$, whose scaled value $\tilde{a}_k/d$ predicts the RMS-scale increment. (c) $\|u_{k+1} - u_k\|_{\text{rms}} s_k$, whose leading-order value is $\|b_\perp(u_k)\|_{\text{rms}}$.

Active radial update. Lemma 2 predicts that, once the model is in the large-scale regime, the RMS scale increment is approximately $a_{\text{rad},k} = \langle u_k, F(H_k) - H_k \rangle / d$, while angular motion shrinks like $1/s_k$. Figure 5 checks this directly by running trained checkpoints for 30 loops on held-out data. The important empirical fact is not the Taylor expansion itself; it is that the learned residual updates enter a stable regime where a persistent radial component accounts for the observed scale motion.

The measured residual update magnitude also separates sharply by readout. Across the diagnostic checkpoints, $\|b\|_{\text{rms}}$ lies in $[0.07, 0.51]$ for raw readouts but in $[20.4, 253.1]$ for normalized readouts. For example, the 44M per-loop+RMSNorm checkpoint has $\tilde{a} \approx 132{,}000$, predicting an RMS-scale increment of about 258 per loop, within 3% of the measured value once the transient settles. The corresponding per-loop+raw checkpoint predicts an increment of only 0.27. The full decomposition is in Appendix G; the normalized-readout checkpoints learn much larger radial residual updates than the raw-readout checkpoints.

H Additional Mitigation Controls

The main text focuses on the interventions that directly test the readout-blind-spot mechanism. We also ran mitigation-style controls that help position the result relative to existing looped-transformer designs.

Exit-weighted objective. An Ouro-style exit-probability-weighted objective keeps RMSNorm readouts but changes how per-loop CE terms are weighted. It reduces neither the qualitative drift pattern nor the radial blind spot: final-loop norms remain in the thousands at both scales ($6{,}470 \pm 2{,}233$ at 44M and $4{,}710 \pm 725$ at 129M). This supports the distinction between changing CE weights and changing what CE can see.

Spectral damping. Spectral damping multiplies the hidden state by a learned factor below 1 between loops. It limits amplification but does not make the readout loss norm-sensitive. In our runs it lowers norms relative to terminal + RMSNorm ($2{,}432 \pm 238$ at 44M and $2{,}973 \pm 16$ at 129M on the stable seeds), but remains far above raw-readout norms. One of three 129M seeds became unstable with $\|H_0\| > 10^5$, so we treat it as less robust than exposing scale through the loss.

Table 15: **Readout normalization removes the local radial gradient.** ρ_k is the fraction of the readout gradient aligned with hidden-state scale at loop k; $g_K = \partial\mathcal{L}/\partial \log \|H_K\|$ is the signed final-loop radial derivative. Raw intermediate readouts restore a scale-sensitive signal; final-only norm restores it only before the final normalized interface.

Model	Readout	$\|H_K\|_2$	ρ_1	ρ_K	g_K
44M	RMSNorm	62,847	5.0×10^{-9}	4.7×10^{-9}	4.0×10^{-10}
44M	Raw	41	6.3×10^{-2}	7.2×10^{-2}	8.2×10^{-2}
44M	Final-only norm	54	6.4×10^{-2}	1.2×10^{-8}	-3.9×10^{-8}
129M	RMSNorm	26,045	8.8×10^{-9}	7.5×10^{-9}	-3.1×10^{-9}
129M	Raw	70	6.0×10^{-2}	4.7×10^{-2}	1.4×10^{-1}
129M	Final-only norm	91	5.9×10^{-2}	9.5×10^{-9}	4.9×10^{-8}

Table 16: **Scale control and exit training are separate design requirements.** The interventions separate whether CE sees scale, whether scale is controlled by an auxiliary loss, or whether scale is removed from the recurrent path.

Intervention	Role in the framework
Scale-hidden CE	
Normalized readout	Hides hidden-state scale from the immediate CE loss.
Per-loop CE through normalized readouts	Trains intermediate prediction exits, but does not by itself make scale visible.
Scale-visible CE	
Raw readout	Makes scale visible to CE by breaking scale invariance.
Explicit scale penalty	
Norm penalty	Adds an auxiliary scale-visible objective while keeping the normalized readout.
Hybrid readout	
Final-only norm	Exposes intermediate loops through raw readouts but keeps a normalized final interface.
Scale-removing recurrence	
Inter-loop normalization	Removes scale from the recurrent input rather than supervising it.

Inter-loop normalization. The released Ouro implementation uses RMSNorm between loop applications. This is a different solution: it caps the scale entering each recurrent update rather than making the output loss scale-sensitive. It is effective, but it changes the loop dynamics and adds normalization inside the recurrent path.

Latent-space losses. In denoising-style looped experiments, per-loop latent losses can prevent drift because they directly train H_k in representation space. This is consistent with our mechanism. Per-loop losses help scale only when the loss itself is scale-sensitive; CE through a normalized readout is not.

I Legacy Halting-Threshold Sensitivity

The recovered threshold sweep applies sequence-level halting to the original 129M per-loop models. At each loop it averages the top-1 versus top-2 logit margin across sequence positions, then halts when that confidence exceeds a calibrated threshold. The threshold is selected to keep PPL within a target budget relative to always running four loops.

The inspected implementation passes one unshuffled loader to calibration and restarts that same loader for evaluation. Consequently, the results below are sensitivity measurements on the calibration sequences, not an evaluation on a disjoint held-out slice. Each recovered CSV contains one checkpoint per condition, rather than a three-seed aggregate. These numbers do not affect the fixed-depth curves in Appendix K, which require no threshold calibration.

Table 17: In-sample halting-threshold sensitivity at 129M. Budgets are relative to fixed $K = 4$ PPL; values are from the recovered individual-checkpoint CSVs.

PPL budget	Raw avg. loops	Raw PPL	RMS avg. loops	RMS PPL
0.5%	2.02	4.8843	1.00	5.3312
1.0%	1.78	4.9061	1.00	5.3312
2.0%	1.36	4.9533	1.00	5.3312

The ordering is unchanged across these calibration budgets. Generalization of the selected thresholds requires a disjoint evaluation slice. The measurements use teacher-forced scoring without KV caching and do not establish autoregressive decoding throughput.

J Terminal-Only Variable-Depth Baselines at 129M

Table 18 reports variable-depth cross-entropy for the terminal-only rows of the 2×2 ablation, which we omit from Table 6 because the main variable-depth comparison concerns per-loop-loss exits. The pattern is informative: terminal + raw has severe degradation at every $K<4$ (CE > 13.8 at $K=3$), and terminal + RMSNorm degrades smoothly from $K=1$ (CE 5.52) to $K=4$ (CE 1.54). Both confirm that per-loop CE is necessary for early exits to produce useful predictions, regardless of readout choice.

Table 18: Variable-depth inference on terminal-only 129M checkpoints (3-seed averages, same variable-depth harness as Table 6), reported as cross-entropy rather than PPL to avoid enormous exponentiated values. Both conditions degrade sharply at $K<K_{\max}$, motivating the per-loop-loss focus in the main text.

Loss	**Readout**	$K=1$ CE	$K=2$ CE	$K=3$ CE	$K=4$ CE
Terminal	RMSNorm	5.52	4.80	3.18	1.54
Terminal	Raw	> 27.6	> 16.1	> 13.8	1.55

K Small-Model Fixed-Depth Quality Curves

The original small-model depth comparisons use a common slice and three training seeds. These fixed-depth results do not depend on threshold calibration. At $K = 1$ (about 80k tokens/s), the 129M raw model has 0.30 lower PPL than the RMSNorm model; at $K = 4$ (about 21k tokens/s), the gap is 0.42. These are teacher-forced scoring measurements under the original small-model recipe.

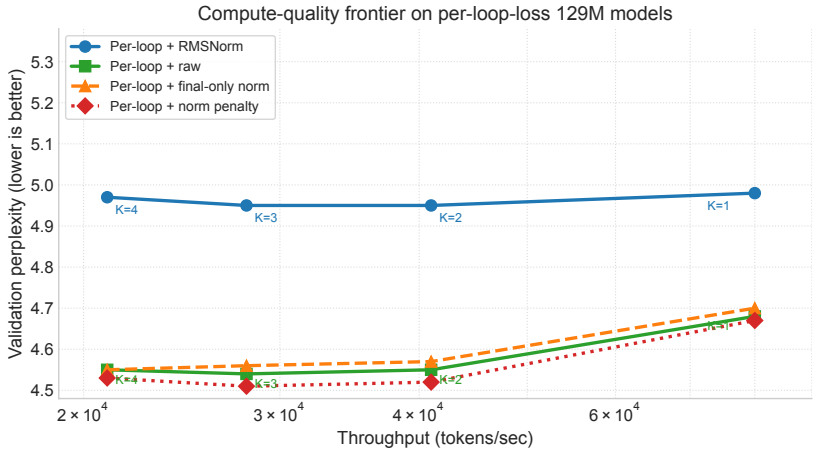

Figure 6: **Compute–quality frontier on per-loop-loss 129M models.** Each curve sweeps $K \in \{1, 2, 3, 4\}$. Raw, final-only norm, and norm penalty occupy the lower-PPL region; RMSNorm is above them at every measured operating point.