DEEPLOOP: DEPTH SCALING FOR LOOPED TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped Transformers scale sequential computation by applying a compact stack of physical blocks for multiple rounds, increasing unrolled depth without increasing stored parameters. This reuse changes the residual-scaling problem: in an untied Transformer, each residual branch receives and applies its own parameter update, whereas in a looped Transformer one shared update aggregates gradients from repeated visits and is read back by those same visits in the next linearized forward pass. We formalize this tied-depth effect through a first-order perturbation bound controlled by a visit-alignment coefficient κ_R. The bound recovers the DeepNorm exponent when visits decorrelate, but in the conservative aligned regime it requires the exponent to increase from $1/4$ to $1/2$ as loop count grows at fixed physical depth. The resulting method, **DeepLoop**, keeps the Post-LN DeepNorm architecture with a normalized branch input and sets $\alpha = (2N)^{1/2}$ and $\beta = (8N)^{-1/2}$ for unrolled depth N. On GPT-style looped language models at GPT-2 small and GPT-2 medium scale, DeepLoop is neutral when no physical block is revisited, improves validation loss at every activated loop count, and improves average downstream accuracy in all but one of the evaluated settings. These results show that stable recurrent depth requires residual scaling rules that account for parameter visits, not only nominal layer count.

1 INTRODUCTION

Depth is one of the most reliable ways to improve Transformer expressivity, but in standard architectures, depth and parameter count grow together: adding a layer also adds a new set of attention and feed-forward weights (Vaswani et al., 2017; Kaplan et al., 2020; Hoffmann et al., 2022). Looped Transformers decouple these axes by applying a stack of K physical blocks for R rounds, yielding unrolled depth $N = KR$ while storing only K blocks (Figure 1). This mechanism connects classical depth-wise sharing in Universal Transformers, ALBERT, and Subformer (Dehghani et al., 2018; Lan et al., 2019; Reid et al., 2021) with recent recurrent-depth and test-time-compute models that spend additional sequential computation without a proportional increase in parameters (Zhu et al., 2026; Fu et al., 2026; Giannou et al., 2023; Yang et al., 2023b; Gatmiry et al., 2024; Geiping et al., 2025; Saunshi et al., 2025). For the looped depth to become a practical scaling axis, however, its residual parameterization must remain stable as the same physical blocks are revisited many times.

The difficulty is that standard residual-scaling analyses are written for untied depth. Residual parameterization is decisive for deep Transformer optimization (Xiong et al., 2020; Nguyen & Salazar, 2019; Liu et al., 2020; Huang et al., 2020; Bachlechner et al., 2021; Wang et al., 2024). DeepNorm, in particular, makes very deep Post-LN Transformers trainable by choosing a skip scale α and a residual-branch initialization gain β so that the first-order effect of parameter updates remains bounded across the $M = 2N$ residual-sublayer visits (Wang et al., 2024), assuming each unrolled residual sublayer owns a distinct parameter tensor and contributes one independent update term.

Weight sharing violates precisely this assumption. When a physical sublayer is visited R times, its optimizer update aggregates the visit-wise gradients from all rounds, and the updated tensor is then reused by all of those visits in the next linearized forward computation: the update is written by many visits and then read by many visits. If visit-wise gradients and sensitivities are nearly orthogonal across rounds, a looped model behaves like untied depth up to constants; if they are coherent, the shared update can acquire an additional factor of R.

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.

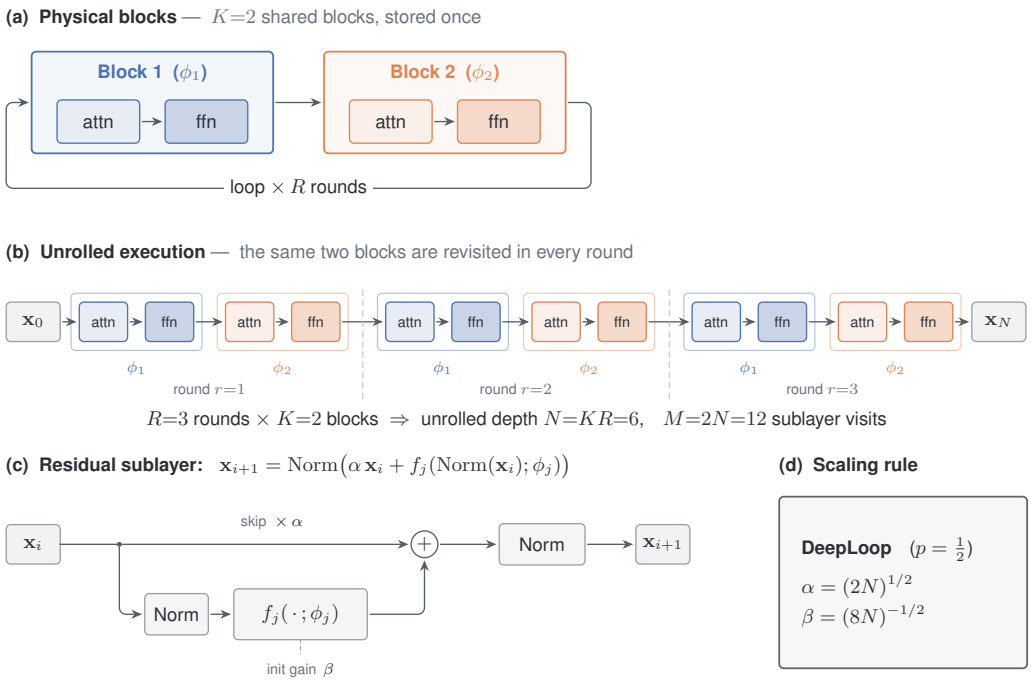


Figure 1: Overview of the DeepLoop framework. **(a)** A looped Transformer stores K=2 physical blocks, each with an attention and an FFN sublayer. **(b)** The K-block stack is applied once per round for R=3 rounds, giving N=KR=6 unrolled blocks and M=2N=12 sublayer visits; the ϕ_j label marks the reused physical block. **(c)** Each visit applies $\mathbf{x}_{i+1}$=Norm$(\alpha\,\mathbf{x}_i + f_j(\mathrm{Norm}(\mathbf{x}_i);\phi_j))$, with the inner normalization fixing the branch-input scale, α scaling the skip connection, and β the per-matrix initialization gain. **(d)** DeepLoop sets α=$(2N)^{1/2}$ and β=$(8N)^{-1/2}$.

This paper makes the tied-depth effect explicit. We introduce a visit-alignment coefficient κ_R and show that the first-order stability condition for a looped Transformer becomes

$$M\kappa_R \left(\frac{\beta}{\alpha}\right)^2 = O(1),$$

rather than the untied DeepNorm condition $M(\beta/\alpha)^2 = O(1)$. For the scaling family $\alpha = (cN)^p$ and $\beta = (dN)^{-p}$, the usual DeepNorm threshold $p = 1/4$ is recovered when visits decorrelate. In the conservative aligned regime, where $\kappa_R = \Theta(R)$ and K is fixed while R grows, the threshold becomes $p = 1/2$.

The resulting method is **DeepLoop**: keep the DeepNorm Post-LN architecture, augmented with the branch-input normalization that is standard in recurrent-depth models (Geiping et al., 2025) and provably inert to the perturbation analysis, and use the loop-aware scaling rule

$$\alpha = (2N)^{1/2}, \qquad \beta = (8N)^{-1/2}.$$

DeepLoop introduces no gates, learned residual coefficients, auxiliary losses, or architecture-specific tuning constants; it is a deterministic residual parameterization for the regime where effective depth is increased by revisiting shared blocks.

We evaluate DeepLoop in controlled GPT-style looped language modeling ablations at GPT-2 small and GPT-2 medium scales. DeepLoop is essentially neutral at $R = 1$, where no physical block is revisited, and consistently improves final validation loss at larger loop counts. The advantage transfers to an eight-task language-model evaluation suite: downstream averages are close at $R = 1$ and, with a single exception at one intermediate loop count, move in favor of DeepLoop as recurrent depth grows. These results support the central prediction of the analysis: residual scaling should depend on how depth is realized, not only on the nominal number of unrolled layers.

Our contributions are:

1. We identify the tied-depth aggregation mechanism absent from untied residual-scaling analyses: a shared update is accumulated across repeated visits and then read through those same visits;
2. We derive a loop-aware first-order perturbation bound with an explicit visit-alignment coefficient κ_R, recovering DeepNorm in the decorrelated regime and yielding a $p = 1/2$ threshold in the aligned fixed-physical-depth regime;
3. We propose DeepLoop, the one-line conservative rule $\alpha = (2N)^{1/2}$, $\beta = (8N)^{-1/2}$ for Post-LN looped Transformers with a normalized branch input;
4. We provide controlled pre-training and downstream ablations at GPT-2 small and medium scale, showing the parameterization becomes useful precisely when the loop count exceeds one, with the exponent probed by a p-sweep (Appendix F).

2 BACKGROUND

A standard depth-N Transformer applies N distinct blocks once. A looped Transformer instead chooses K physical blocks and applies them for R rounds,

$$\mathbf{x}_{r,k+1} = B_k(\mathbf{x}_{r,k}; \phi_k), \qquad k = 1, \dots, K, \quad r = 1, \dots, R, \qquad \mathbf{x}_{r+1,1} = \mathbf{x}_{r,K+1}, \tag{2.1}$$

so the effective depth is $N = KR$ while the block parameters $\phi_1, \dots, \phi_K$ are stored only once. The Universal Transformer (Dehghani et al., 2018) corresponds to the fully recurrent extreme in which the same transition is reused across depth; ALBERT (Lan et al., 2019) similarly ties Transformer parameters across layers. The regime studied here keeps K fixed or small and increases R, thereby increasing test-time compute and unrolled depth without increasing the number of physical blocks.

Let g_ℓ denote one residual sublayer, either attention or an MLP. The two common normalization placements are

$$\text{Pre-LN:} \quad \mathbf{x}_{\ell+1} = \mathbf{x}_\ell + g_\ell(\mathrm{Norm}(\mathbf{x}_\ell)), \qquad \text{Post-LN:} \quad \mathbf{x}_{\ell+1} = \mathrm{Norm}\left(\mathbf{x}_\ell + g_\ell(\mathbf{x}_\ell)\right).$$

Pre-LN improves optimization stability in deep Transformers, while Post-LN can preserve a more expressive residual stream (Xiong et al., 2020; Nguyen & Salazar, 2019). Recent recurrent-depth models additionally normalize the branch input inside a post-normalized residual, a sandwich placement that combines the two schemes (Geiping et al., 2025); Section 3.1 adopts this placement and shows that it leaves the perturbation analysis unchanged. DeepNorm (Wang et al., 2024) modifies the Post-LN residual path by scaling the skip connection before normalization:

$$\mathbf{x}_{\ell+1} = \mathrm{Norm}\left(\alpha\, \mathbf{x}_\ell + g_\ell(\mathbf{x}_\ell; \theta_\ell)\right). \tag{2.2}$$

For an encoder-only or decoder-only Transformer with N blocks and $M = 2N$ residual sublayer applications, DeepNorm sets

$$\alpha = (2N)^{1/4}, \qquad \beta = (8N)^{-1/4}, \tag{2.3}$$

Here β is an initialization gain, not an additional runtime multiplier on the residual stream: it rescales the DeepNorm-designated residual-branch matrices, per matrix, at initialization (Appendix B restates the per-matrix definition). The quantity that enters the perturbation argument is the update-to-residual ratio $\beta_{\mathrm{DN}}/\alpha_{\mathrm{DN}} = (8N)^{-1/4}/(2N)^{1/4} = 1/(2\sqrt{N})$. The useful way to summarize the DeepNorm calculation is not a bound on $\alpha^2\beta$, but the first-order update condition

$$M\left(\frac{\beta}{\alpha}\right)^2 = \Theta(1). \tag{2.4}$$

Indeed, substituting the DeepNorm ratio gives $2N(\beta/\alpha)^2 = 1/2$.

In a non-shared depth-N Transformer, each sublayer parameter receives one gradient contribution per input; in a looped Transformer the same physical parameter is visited once per round, so its gradient is a sum over R visits that is then read back by all R visits in the unrolled computation. DeepLoop is the corresponding conservative correction to Eq. (2.4): it chooses a smaller update-to-residual ratio β/α so that the first-order effect of the tied parameter update remains bounded after all R visits are unrolled.

3 DEEPLOOP TRANSFORMER

A looped Transformer reuses the same K physical blocks for R rounds. Its unrolled depth is $N = KR$, and the number of residual-sublayer visits is $M = 2N$. DeepLoop keeps the DeepNorm architecture with a normalized branch input and uses the exponent $p = 1/2$ rather than $p = 1/4$ in the fixed-physical-depth, increasing-loop-count regime. The derivation separates two facts: RMSNorm exposes the residual branch through a factor $1/\alpha$, and parameter tying changes the first-order update sum from M terms to $M\kappa_R$ terms, where κ_R measures alignment across loop visits.

3.1 SETUP: LOOPED POST-NORMALIZED BLOCK

Each physical block contains two residual sublayers: attention and MLP. Let $j = (k, s)$ index a physical sublayer, with $k \in \{1, \dots, K\}$ and $s \in \{\mathrm{attn}, \mathrm{ffn}\}$, and let $i = (r, j)$ denote its r-th unrolled visit. We write $J = 2K$ for the number of physical residual sublayers and $M = JR = 2KR = 2N$ for the number of unrolled visits. DeepLoop uses the post-normalized sandwich block

$$\mathbf{x}_{i+1} = \mathrm{Norm}\left(\alpha\, \mathbf{x}_i + f_j(\mathrm{Norm}(\mathbf{x}_i); \phi_j)\right), \qquad i = 1, \dots, M, \tag{3.1}$$

where the same physical parameter ϕ_j is reused for all $r = 1, \dots, R$, and both normalizations are RMSNorm. The loop entry is normalized by an input-embedding RMSNorm and every later state is an output of the outer normalization, so at initialization, where the perturbation analysis is conducted and all normalization gains equal one, the stream entering each visit has unit RMS. On this stream the inner normalization acts as the identity, and Lemma A.2 in Appendix A shows that the analysis below applies verbatim to Eq. (3.1) with or without it. Its role is architectural rather than analytical: it presents every visit with a unit-RMS branch input, whatever the residual-stream scale does during training, which is the regime under which the visit-wise constants of Assumption 3.1 below are calibrated. The gain β is a per-matrix initialization gain applied to the residual-branch matrices specified by DeepNorm, not the standard deviation of the whole sublayer output, since a branch may contain multiple scaled matrices: for each physical sublayer j with DeepNorm-scaled matrix set $\mathcal{S}_j$, DeepLoop initializes

$$W_{j,q}^{(0)} = \beta_{\mathrm{DL}}\, \widetilde{W}_{j,q}^{(0)}, \qquad q \in \mathcal{S}_j, \qquad \beta_{\mathrm{DL}} = (8N)^{-1/2}. \tag{3.2}$$

The forward recurrence of Eq. (3.1) then uses these initialized parameters at every visit; β is not re-applied as a separate multiplicative factor on each visit. With $\alpha_{\mathrm{DL}} = (2N)^{1/2}$, the DeepLoop update-to-residual ratio is

$$\frac{\beta_{\mathrm{DL}}}{\alpha_{\mathrm{DL}}} = \frac{(8N)^{-1/2}}{(2N)^{1/2}} = \frac{1}{4N}. \tag{3.3}$$

Assumption 3.1 (DeepNorm-scale local sensitivity)**.** For each unrolled visit $i = (r, j)$, the norm of the linearized output map from a residual-branch parameter perturbation at that visit and the norm of the corresponding visit-wise effective update are both $O(\beta/\alpha)$.

Assumption 3.1 is the local scaling condition used in the DeepNet perturbation argument. Constants depending on width, normalization gain, attention heads, learning rate, or optimizer preconditioning are absorbed into the $O(\cdot)$ notation; by Lemma A.2, on the unit-RMS stream of the initialized model the inner normalization of Eq. (3.1) contributes to these constants only factors of operator norm one. The looped analysis below changes only how the visit-wise terms are aggregated when parameters are tied.

3.2 FORWARD STABILITY AND THE DEPTH-UNTIED DEEPNORM BOUND

RMSNorm restores unit RMS at every sublayer, so the forward signal scale is not the binding constraint; the relevant constraint is the first-order sensitivity of the final output to a residual-branch parameter update. A direct expansion of $\mathrm{RMSNorm}(\alpha\mathbf{x}+\mathbf{z})$ around $\mathrm{RMS}(\mathbf{z})/\alpha \to 0$ shows that the residual branch enters the normalized direction through a factor of $1/\alpha$ (Lemma A.1 in Appendix A), which is why the relevant DeepNorm ratio is β/α rather than β alone; the inner normalization complements this mechanism on the input side by pinning the branch input to unit RMS at every visit, so the branch-output scale is governed by the β-scaled matrices alone.

The DeepNet perturbation argument bounds the first-order output change $\Delta F = F(\mathbf{x}; \theta + \delta\theta) - F(\mathbf{x}; \theta)$ by summing visit-wise sensitivities. Here *depth-untied* means that residual-sublayer parameters are not shared across unrolled depth. This is orthogonal to the standard tying of input and output token embeddings: tied embeddings, when used, are outside the $M = 2N$ residual-sublayer count and are not among the residual-branch matrices scaled by β. Under Assumption 3.1, each visit contributes an output sensitivity $O(\beta/\alpha)$ and an effective update $O(\beta/\alpha)$, so summing $M = 2N$ depth-untied residual-sublayer visits gives

$$\|\Delta F\| \leq C' M \left(\frac{\beta}{\alpha}\right)^2, \tag{3.4}$$

and a sufficient first-order stability condition is

$$M \left(\frac{\beta}{\alpha}\right)^2 = O(1). \tag{3.5}$$

The DeepNorm choice $\alpha = (2N)^{1/4}$ and $\beta = (8N)^{-1/4}$ satisfies $M(\beta/\alpha)^2 = 1/2$. The formal proposition statement and proof are Proposition A.3 in Appendix A.

3.3 Loop correction: aligned visits require $p = 1/2$

Depth-wise residual-sublayer weight sharing changes Eq. (3.4). For a physical sublayer j, let $G_{r,j}$ be the visit-wise effective update that would be applied if visit (r, j) had its own depth-untied residual-branch parameter, and let $U_{r,j}$ be the corresponding linearized output-sensitivity operator. With tied residual-sublayer parameters, the update to ϕ_j is proportional to $\sum_{r=1}^{R} G_{r,j}$, and the output perturbation reads this same update through all visits. The first-order tied perturbation therefore has the schematic double-sum form

$$\Delta F_{\text{tied}} = -\eta \sum_{j=1}^{J} \left(\sum_{r=1}^{R} U_{r,j}\right) \left(\sum_{t=1}^{R} G_{t,j}\right) + O(\eta^2), \tag{3.6}$$

where tensor contractions and optimizer-dependent constants are absorbed into $U_{r,j}$ and $G_{r,j}$.

Define the visit-alignment coefficient

$$\kappa_R := \max_{j \in \{1,\ldots,J\}} \frac{\left\|\sum_{r=1}^{R} U_{r,j}\right\| \left\|\sum_{r=1}^{R} G_{r,j}\right\|}{R C_U C_G (\beta/\alpha)^2}, \tag{3.7}$$

where $\|U_{r,j}\| \leq C_U \beta/\alpha$ and $\|G_{r,j}\| \leq C_G \beta/\alpha$. The triangle inequality gives $0 \leq \kappa_R \leq R$. Independent or nearly orthogonal visits give $\kappa_R = O(1)$, while fully aligned visits give $\kappa_R = \Theta(R)$.

Substituting the definition of κ_R from Eq. (3.7) into Eq. (3.6) and applying submultiplicativity gives the looped version of the DeepNorm bound,

$$\|\Delta F\| \leq C'' M \kappa_R \left(\frac{\beta}{\alpha}\right)^2, \tag{3.8}$$

so a sufficient tied-depth stability condition is

$$M \kappa_R \left(\frac{\beta}{\alpha}\right)^2 = O(1). \tag{3.9}$$

In the worst-case aligned regime $\kappa_R = \Theta(R)$, this reduces to $MR(\beta/\alpha)^2 = O(1)$. The formal proposition and the aligned-case corollary are Proposition A.4 and Corollary A.5 in Appendix A.

Now consider the family $\alpha = (cN)^p$, $\beta = (dN)^{-p}$ with constants $c, d > 0$, whose update-to-residual ratio is $\beta/\alpha = (cd)^{-p} N^{-2p}$.

Proposition 3.2 (Exponent threshold)**.** Assume K is fixed, $N = KR$, $M = 2N$, and $\kappa_R = \Theta(R^\gamma)$ for some $\gamma \in [0, 1]$. For the scaling family $\alpha = (cN)^p$, $\beta = (dN)^{-p}$, the tied-depth condition

$$M \kappa_R (\beta/\alpha)^2 = O(1)$$

holds uniformly as $R \to \infty$ if and only if

$$p \geq \frac{1 + \gamma}{4}.$$

Proposition 3.2 shows where the new exponent comes from. When visits decorrelate, $\gamma = 0$, and the DeepNorm threshold $p = 1/4$ is recovered. In the conservative aligned case, $\gamma = 1$, the threshold is $p = 1/2$: at fixed K, $R = N/K$, and DeepNorm's $p = 1/4$ leaves a residual growth of $\Theta(R)$ in the aligned shared-loop bound. The DeepLoop choice is

$$\alpha = (2N)^{1/2}, \qquad \beta = (8N)^{-1/2}.$$

We keep the DeepNorm constants $(c, d) = (2, 8)$ for a two-sublayer decoder block and change only the exponent. Equivalently, DeepLoop uses the same per-matrix definition as Eq. (3.2) with $\beta = \beta_{\mathrm{DL}}$. With this choice $\beta/\alpha = 1/(4N)$ (Eq. (3.3)), and the worst-case aligned tied-depth bound evaluates to $MR(\beta/\alpha)^2 = 1/(8K) = O(1)$ at fixed physical depth K; the intermediate algebra is recorded in Appendix B. Proofs of Proposition A.4 and Proposition 3.2 are in Appendix A. Appendix F sweeps p over a single grid at R=3 and finds an empirical training-stability transition that brackets this p=1/2 prediction.

Table 4 in Appendix B contrasts the two scaling rules line by line. DeepLoop is therefore a one-line conservative correction for tied depth: increase the exponent from $p = 1/4$ to $p = 1/2$, with no gates, learnable residual scalars, or additional tuning constants. The correction is not a claim that every looped model empirically attains worst-case alignment; rather, $p = 1/2$ is the minimal exponent for uniform boundedness under aligned visit-wise contributions at fixed physical depth.

4 Application to Hierarchical Recurrent Reasoners

The DeepLoop derivation in Section 3 treats a single looped block visited for R rounds. Recent recurrent-reasoning architectures combine looping with two further structural choices: a hierarchy of recurrent modules executed at different rates, and training-time gradient truncation across the outer recurrence (Wang et al., 2025; Geiping et al., 2025). The Hierarchical Reasoning Model (Wang et al., 2025) instantiates both: a high-level module $\mathcal{H}$ with K_H physical blocks and a low-level module $\mathcal{L}$ with K_L physical blocks (each block again two residual sublayers) are unrolled jointly for C outer cycles, with $\mathcal{L}$ iterating C_L times before each single update of $\mathcal{H}$: cycle c applies $\mathbf{z}_L \leftarrow \mathcal{L}^{(C_L)}(\mathbf{z}_L, \mathbf{z}_H)$ and then $\mathbf{z}_H \leftarrow \mathcal{H}(\mathbf{z}_H, \mathbf{z}_L)$, for $c = 1, \ldots, C$, and gradients are computed through the last outer cycle only. The blocks use the post-normalized recurrence of Eq. (3.1) without the inner branch normalization, the form for which the propositions of Appendix A are stated directly. This section shows that the framework of Section 3 extends to this regime by replacing the unrolled visit count M with a gradient-visible visit count and by tracking per-module visit counts separately; we introduce no new optimization machinery and no new variant of DeepLoop. Full bookkeeping, derivations, and additional structural remarks are collected in Appendix C.

Counting attention and MLP residual sublayers, the forward pass unrolls $M = 2\,C\,(K_H + C_L K_L)$ visits, while the truncated backward pass sees only the last outer cycle, with gradient-visible visit count

$$M_{\mathrm{g}} := 2\,(K_H + C_L K_L), \qquad M = C\,M_{\mathrm{g}}. \tag{4.1}$$

On the gradient-visible cycle, each physical $\mathcal{H}$-sublayer is visited once and each physical $\mathcal{L}$-sublayer C_L times (Eq. (C.2)). Repeating the argument leading to Eq. (3.8), with the visit-index sums restricted to the gradient-visible cycle, gives the two-module loop-aware bound

$$\|\Delta F\| \le C'' \Big[J_H\, R_{\mathrm{g}}^{(\mathcal{H})}\, \kappa_{\mathrm{g}}^{(\mathcal{H})} + J_L\, R_{\mathrm{g}}^{(\mathcal{L})}\, \kappa_{\mathrm{g}}^{(\mathcal{L})} \Big] \left(\frac{\beta}{\alpha}\right)^2, \tag{4.2}$$

where $J_H = 2K_H$ and $J_L = 2K_L$ count the physical residual sublayers, $R_{\mathrm{g}}^{(\mathcal{H})} = 1$ and $R_{\mathrm{g}}^{(\mathcal{L})} = C_L$ are the per-module gradient-visible round counts, and $\kappa_{\mathrm{g}}^{(\mathcal{H})}, \kappa_{\mathrm{g}}^{(\mathcal{L})}$ are the visit-alignment coefficients of Eq. (3.7) restricted to each module's gradient-visible visits. Because $R_{\mathrm{g}}^{(\mathcal{H})} = 1$, the coefficient $\kappa_{\mathrm{g}}^{(\mathcal{H})}$ collapses to a constant, so the truncation places $\mathcal{H}$ in the untied-DeepNorm regime even though $\mathcal{H}$ is forward-visited C times. The inner $\mathcal{L}$-iterate sum gives $\kappa_{\mathrm{g}}^{(\mathcal{L})} \in [0, C_L]$ by the triangle inequality. Substituting these facts yields the sufficient stability condition

$$M_{\mathrm{g}}\, \bar{\kappa}_{\mathrm{g}} \left(\frac{\beta}{\alpha}\right)^2 = O(1), \qquad \bar{\kappa}_{\mathrm{g}} := \frac{J_H + J_L\, C_L\, \kappa_{\mathrm{g}}^{(\mathcal{L})}}{M_{\mathrm{g}}}, \tag{4.3}$$

which has the same structural form as Eq. (3.9) with $M \to M_{\mathrm{g}}$ and $\kappa_R \to \bar{\kappa}_{\mathrm{g}}$; the bound also decomposes per module and admits asymmetric per-module (α, β) assignments (Appendix C).

Let $N_{\mathrm{g}} := M_{\mathrm{g}}/2$ denote the gradient-visible block-equivalent depth. Combining Eq. (4.3) with the same two-sublayer scaling family used above, $\alpha = (cN_{\mathrm{g}})^p$, $\beta = (dN_{\mathrm{g}})^{-p}$, gives, up to constant factors,

$$M_{\mathrm{g}}\, \bar{\kappa}_{\mathrm{g}}\, (cd)^{-2p}\, N_{\mathrm{g}}^{-4p} = \Theta\big((cd)^{-2p}\, \bar{\kappa}_{\mathrm{g}}\, M_{\mathrm{g}}^{1-4p}\big)\,, \tag{4.4}$$

and the asymptotic threshold depends on how $\bar{\kappa}_{\mathrm{g}}$ and M_{g} co-scale, mirroring Proposition 3.2. Decorrelated inner cycles, or growth of M_{g} through the physical depths K_H, K_L at fixed C_L, give the untied threshold $p \geq 1/4$; aligned inner cycles ($\kappa_{\mathrm{g}}^{(\mathcal{L})} = \Theta(C_L)$) with depth grown through C_L at fixed physical depth give $p \geq 1/2$ (Appendix C).

For a hierarchical recurrent reasoner as actually built, these two limits are a binding constraint and a slack one. The truncated once-per-step $\mathcal{H}$ module requires only $p \geq 1/4$. The $\mathcal{L}$ module realizes the model's effective depth by revisiting its shared blocks at fixed physical depth, and weight tying is adopted precisely so that those revisits implement the *same* operation, aligning their visit-wise gradients and sensitivities; it therefore requires $p \geq 1/2$. A single shared residual exponent must satisfy the more demanding module, so the framework predicts $p = 1/2$ for a hierarchical recurrent reasoner, the same loop-aware exponent as the single-module backbone. As in Section 3, this is a worst-case prescription; we do not claim that the alignment $\kappa_{\mathrm{g}}^{(\mathcal{L})} = \Theta(C_L)$ is empirically attained. Section 5 tests the prediction on ARC-AGI.

5 Experiments

5.1 Validation loss

We compare DeepLoop against a baseline that already incorporates the same looped residual-block sharing, tied input-output embeddings, and an input-embedding RMSNorm. The baseline (`base`) is the standard pre-normalized looped Transformer: each sublayer applies $\mathbf{x} \leftarrow \mathbf{x} + g(\mathrm{Norm}(\mathbf{x}))$ with standard initialization and no residual scaling. The DeepLoop variant instead uses the post-normalized sandwich block of Eq. (3.1) with the scaling rule enabled on every block, so the comparison is between the modern pre-LN default and the complete DeepLoop parameterization; the stability effect of the exponent alone, within the fixed sandwich block, is probed by the p-sweep of Appendix F. Both configurations share the same attention and MLP backbone, optimizer, data pipeline, and random seed, and the loop count $R \in \{1, 3, 5, 7\}$ moves between rows of Table 1.

The GPT-2 small runs use a GPT-MHA-RoPE model trained on FineWeb-Edu for 50B tokens (100K optimizer steps, context length 1024, global batch size 480) on 4×H200 141GB GPUs; the GPT-2 medium runs enlarge the hidden size from 768 to 1024 and double the layers from 12 to 24, with the same data and schedule on 8×H200. We report final-checkpoint validation cross-entropy for both scales in Table 1.

Table 1: Final validation loss at step 100,000 on FineWeb-Edu 50BT for the GPT-2 small and GPT-2 medium backbones, varying loop count $R \in \{1, 3, 5, 7\}$. Both methods share tied input-output embeddings and an input-embedding RMSNorm; the baseline uses the standard pre-LN block. Bold marks the best cell per column per scale; ties within 0.002 nats are bolded on both rows.

Method	$R = 1$	$R = 3$	$R = 5$	$R = 7$
GPT-2 small backbone				
baseline (pre-LN)	**2.8627**	2.8077	2.7910	2.7700
DeepLoop	**2.8631**	**2.7917**	**2.7679**	**2.7514**
Δ (DeepLoop − base)	$+0.0004$	-0.0160	-0.0231	-0.0186
GPT-2 medium backbone				
baseline (pre-LN)	**2.6253**	2.5779	2.5640	2.5558
DeepLoop	**2.6264**	**2.5627**	**2.5444**	**2.5280**
Δ (DeepLoop − base)	$+0.0011$	-0.0153	-0.0196	-0.0278

Figure 2 in Appendix D visualizes the same numbers as Table 1. The two methods coincide within noise at R=1, and DeepLoop improves over the baseline at every $R \geq 3$ at both scales, with the

medium-scale gap widening monotonically to -0.028 nats at R=7 and the small-scale gap peaking at -0.023 nats at R=5. Both methods continue to improve monotonically with R. Multi-seed runs would be needed to quantify run-to-run variance.

5.2 Downstream evaluation on the `lm-evaluation-harness` suite

To test whether the validation-loss advantage transfers to downstream task accuracy, we evaluate the eight medium-scale checkpoints from §5.1 on the eight-task zero- and one-shot suite used by Zhang et al. (2026): `arc_challenge`, `arc_easy`, `hellaswag`, `openbookqa`, `piqa`, `sciq`, `social_iqa`, and `winogrande`. Evaluation uses the `lm-evaluation-harness` (Gao et al., 2024) with its default `acc` metric (not `acc_norm`), the GPT-2 tokenizer, a context length of 1024 (matching training), and bf16 inference. The reported *Avg* column is the unweighted arithmetic mean of the eight per-task accuracies.

Table 2 reports the eight-task average for every (method, R, shot) cell at the medium scale. The full per-task breakdown is Table 5 in Appendix D, and Figure 3 there breaks the 1-shot column out task-by-task. The corresponding GPT-2 small downstream results, which exhibit the same qualitative pattern, are deferred to Appendix E.

Table 2: Eight-task average downstream accuracy (%) for the GPT-2 medium backbone on FineWeb-Edu 50BT, varying loop count $R \in \{1, 3, 5, 7\}$. Metric is `acc`; each entry is the unweighted arithmetic mean across the eight tasks. Bold marks the better cell per column within each shot setting; both methods are tied within noise at R=1. Per-task columns are in Table 5 (Appendix D).

Avg (%)	R=1	R=3	R=5	R=7
0-shot				
baseline (pre-LN)	50.86	52.50	52.61	52.95
DeepLoop	50.85	**52.93**	**53.67**	**53.88**
1-shot				
baseline (pre-LN)	51.69	53.10	**54.42**	54.62
DeepLoop	51.53	53.42	54.19	**55.20**

Two takeaways. First, the qualitative picture from the small-scale ablation (Appendix Table 6) carries over to the larger backbone: the two methods are tied at R=1, DeepLoop opens the gap at R=3, and the best Avg cell is DeepLoop R=7 in both shot settings, headlined by $55.20\,\%$ on 1-shot. The one exception is the 1-shot Avg at R=5, where the base leads by 0.23 points before DeepLoop recovers a $+0.58$-point lead at R=7. Second, at R=7 DeepLoop beats the same-R baseline on seven of the eight individual tasks under both shot settings (only PIQA goes the other way). The same caveat about single-seed run-to-run variance from §5.1 applies; we see no qualitative disagreement with the small-scale ablation, only a stronger depth signal.

5.3 Reasoning evaluation on ARC-AGI

To test the p=1/2 prediction of Section 4 on a hierarchical recurrent reasoner, we apply the DeepLoop scaling rule with $p = 1/2$ to the Hierarchical Reasoning Model (HRM) (Wang et al., 2025) and evaluate on ARC-AGI-1 (Chollet, 2019). The only change from the published HRM is the residual parameterization: every reasoning block uses $\alpha = (2N_{\mathrm{g}})^{1/2}$ and $\beta = (8N_{\mathrm{g}})^{-1/2}$ with $M_{\mathrm{g}} = 2\,(K_H + C_L K_L) = 24$ gradient-visible residual-sublayer visits for the published configuration (K_H=K_L=4 blocks per module, C_L=2 inner iterations) and $N_{\mathrm{g}} = M_{\mathrm{g}}/2 = 12$ block-equivalent depth (Eq. 4.1); the backbone, the adaptive-computation halting, the optimizer (AdamATan2), the data pipeline, and the 100K-epoch schedule are held fixed. Baseline and DeepLoop runs train on an identical, hash-verified `arc-aug-1000` build and are scored by the same count-first voting protocol on $N = 400$ evaluation puzzles. As a harness check, the published HRM ARC-2 checkpoint scores 5.00% (K=2) on this pipeline, matching the released value.

Table 3 reports voted accuracy at voting budgets $K \in \{1, 2, 10, 100, 1000\}$. DeepLoop improves the paper-protocol two-vote accuracy from 36.50% to 39.75% ($+3.25$ pp) and improves every column of the voting ladder. A four-seed control places the per-seed K=2 standard deviation at ≈ 0.5 pp, so the $+3.25$ pp gain is roughly a 6σ effect rather than a seed draw. The prediction of Section 4 is therefore

borne out: a single residual-scaling rule, $p = 1/2$, is the correct setting for both the looped language model and the hierarchical recurrent reasoner.

Table 3: Voted accuracy (%) on ARC-AGI-1 for the vanilla Hierarchical Reasoning Model and the same model with the DeepLoop residual scaling ($p = 1/2$), at voting budgets K. Bold marks the better cell per column; the K=2 column is the paper-protocol headline metric.

Method	K=1	K=2	K=10	K=100	K=1000
Vanilla HRM	31.50	36.50	41.50	47.50	50.75
DeepLoop (p=1/2)	**35.50**	**39.75**	**44.25**	**49.75**	**51.50**
Δ	+4.00	+3.25	+2.75	+2.25	+0.75

6 Related Work

Looped and universal Transformers. Universal Transformers (Dehghani et al., 2018) and ALBERT (Lan et al., 2019) introduced depth-wise parameter sharing as a way to increase effective computation without increasing parameter count. Later work studied partial sharing (Reid et al., 2021), looped Transformers for in-context learning and algorithmic computation (Giannou et al., 2023; Yang et al., 2023b; Gatmiry et al., 2024), and recurrent-depth language models that trade inference compute for quality (Geiping et al., 2025). DeepLoop is complementary to these architectures: within the post-normalized sandwich block of Eq. (3.1), it prescribes the residual and initialization scales used when a physical block is revisited.

Adaptive computation and iterative reasoning. Looped depth can also be viewed as a differentiable test-time-compute mechanism. Adaptive Computation Time (Graves, 2016), PonderNet (Banino et al., 2021), and recurrent networks for algorithmic extrapolation (Schwarzschild et al., 2021; Bansal et al., 2022) all exploit repeated application of shared computation. Recent language-modeling work similarly adds latent or recurrent computation (Goyal et al., 2023; Hao et al., 2024; Saunshi et al., 2025). Our analysis targets the stability of this repeated shared computation.

Residual scaling and parameterization. Transformer stability has been improved through normalization placement (Xiong et al., 2020; Nguyen & Salazar, 2019), initialization and residual scaling (Zhang et al., 2019; Huang et al., 2020; De & Smith, 2020; Bachlechner et al., 2021; Liu et al., 2020), and DeepNorm/DeepNet (Wang et al., 2024), which derives $\alpha = (2N)^{1/4}$ and $\beta = (8N)^{-1/4}$ for untied Post-LN Transformers. Width-and-depth parameterizations such as μP and Depth-μP prescribe complementary scalings for feature learning and hyperparameter transfer (Yang & Hu, 2021; Yang et al., 2021; 2023a; Bordelon et al., 2023). These analyses do not account for the extra correlation introduced when the same residual-branch parameters are reused across loop visits; DeepLoop is the loop-specific correction to the DeepNorm exponent.

7 Conclusion

Looped Transformers turn depth into a controllable compute resource, but repeated parameter reuse changes the residual-scaling problem. We captured this tied-depth effect with a visit-alignment coefficient κ_R, recovering the DeepNorm exponent for decorrelated visits and identifying $p = 1/2$ as the conservative exponent for aligned visits at fixed physical depth.

DeepLoop implements this correction in the post-normalized sandwich block of Eq. (3.1) with a one-line scaling rule, $\alpha = (2N)^{1/2}$ and $\beta = (8N)^{-1/2}$. Empirically, it is neutral when no physical block is revisited and improves looped GPT-style language models as recurrent depth increases, with gains in validation loss and downstream accuracy. A p-sweep at R=3 further places the empirical stability transition in the vicinity of $p = 1/2$.

Future work should measure κ_R or cross-round gradient alignment directly, test whether the same boundary holds at larger scale or under alternative parameterizations, and explore whether training can encourage decorrelated visits to safely use less conservative exponents.

AI USE STATEMENT

In this work, we used generative AI tools as coding assistants for experiment infrastructure, evaluation scripts, and plotting, and for light editing of the manuscript text (grammar and phrasing). All AI-assisted code was verified and tested for correctness by the authors, and all AI-assisted text was reviewed and edited by the authors. Generative AI tools were not used to generate the research ideas or the scientific claims of this work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Complete proofs of all theoretical claims, together with the assumptions they rely on, are given in Appendix A. The architecture, the DeepLoop scaling rule, and the training configuration of every experimental cell are specified in Sections 3 and 5, and the compute budget of each cell is reported in Appendix G. Evaluation uses the public `lm-evaluation-harness` suite and the public ARC-AGI benchmark with the settings described in Sections 5.2 and 5.3.

REFERENCES

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

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

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Extrapolation without overthinking. *Advances in Neural Information Processing Systems*, 35:20232–20242, 2022.

Blake Bordelon, Lorenzo Noci, Mufan Bill Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. *arXiv preprint arXiv:2309.16620*, 2023.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Soham De and Sam Smith. Batch normalization biases residual blocks towards the identity function in deep networks. *Advances in Neural Information Processing Systems*, 33:19964–19975, 2020.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Hengyu Fu, Tianyu Guo, Zixuan Wang, Hanlin Zhu, Jason D. Lee, Jiantao Jiao, Stuart Russell, and Song Mei. Discoloop: Looping discrete embeddings and continuous hidden states for multi-hop reasoning, 2026. URL `https://arxiv.org/abs/2607.00341`.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

Khashayar Gatmiry, Nikunj Saunshi, Sashank J Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? *arXiv preprint arXiv:2410.08292*, 2024.

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.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023.

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

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pp. 4475–4483. PMLR, 2020.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

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

Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. In *Proceedings of the 16th international conference on spoken language translation*, 2019.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 4081–4090, 2021.

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.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. *Advances in Neural Information Processing Systems*, 34:6695–6706, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

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

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

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

Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pp. 11727–11737. PMLR, 2021.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks. *arXiv preprint arXiv:2310.02244*, 2023a.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. *arXiv preprint arXiv:2311.12424*, 2023b.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

Yifan Zhang, Yifeng Liu, Mengdi Wang, and Quanquan Gu. Deep delta learning. *arXiv preprint arXiv:2601.00417*, 2026.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models, 2026. URL https://arxiv.org/abs/2510.25741.

Appendix

A Proofs **14**

B Extended derivations and comparisons **16**

B.1 Per-matrix use of the DeepNorm initialization gain 16

B.2 Aligned tied-loop bound for the scaling family 16

B.3 Side-by-side comparison with DeepNorm 16

C Hierarchical recurrent reasoners: extended analysis **17**

C.1 Setup and visit bookkeeping 17

C.2 One-step gradient approximation and the per-module condition 17

C.3 Limit cases for the exponent threshold 17

C.4 Forward signal at the loop entry 18

D Medium-scale experimental detail **19**

E Small-scale downstream evaluation **21**

F Empirical p-sweep at fixed loop count **23**

G Compute resources **24**

A PROOFS

Lemma A.1 (RMSNorm exposes the residual branch through $1/\alpha$)**.** Let $\mathrm{RMS}(x)^2 = d^{-1}\|x\|^2$, assume $\mathrm{RMS}(x) = 1$, and define $\mathcal{R}(y) = y/\mathrm{RMS}(y)$. If $\mathrm{RMS}(z)/\alpha \le c < 1$, then

$$\mathcal{R}(\alpha x + z) = x + \frac{z - \langle x, z\rangle_d x}{\alpha} + O\left(\frac{\mathrm{RMS}(z)^2}{\alpha^2}\right), \qquad \langle x, z\rangle_d := d^{-1}x^\top z. \tag{A.1}$$

Proof of Lemma A.1. Write $m = \langle x, z\rangle_d$ and $\delta^2 = \mathrm{RMS}(z)^2$. Since $|m| \le \delta$ and $\delta/\alpha \le c < 1$, the quantity below is bounded away from zero:

$$\mathrm{RMS}(\alpha x + z) = \alpha\left(1 + 2m/\alpha + \delta^2/\alpha^2\right)^{1/2}.$$

A first-order Taylor expansion of the inverse square root around one gives

$$\left(1 + 2m/\alpha + \delta^2/\alpha^2\right)^{-1/2} = 1 - m/\alpha + O(\delta^2/\alpha^2),$$

where the remainder is uniform for $\delta/\alpha \le c$. Multiplying by $(\alpha x + z)/\alpha$ yields

$$\frac{\alpha x + z}{\mathrm{RMS}(\alpha x + z)} = x + \frac{z - mx}{\alpha} + O(\delta^2/\alpha^2),$$

which is Eq. (A.1). □

Lemma A.2 (Inner normalization is inert on a post-normalized stream)**.** Let $\mathcal{R}(y) = y/\mathrm{RMS}(y)$ as in Lemma A.1. *(i)* If $\mathrm{RMS}(x) = 1$, then $\mathcal{R}(x) = x$. *(ii)* For any x with $\mathrm{RMS}(x) > 0$ and $d \ge 2$, the Jacobian of $\mathcal{R}$ is

$$\partial\mathcal{R}(x) = \frac{1}{\mathrm{RMS}(x)}\left(I - \hat{x}\hat{x}^\top\right), \qquad \hat{x} := x/\|x\|,$$

an orthogonal projection scaled by $1/\mathrm{RMS}(x)$, so its operator norm is $1/\mathrm{RMS}(x)$; in particular it equals one when $\mathrm{RMS}(x) = 1$.

Proof of Lemma A.2. Part *(i)* is immediate from the definition. For *(ii)*, write $\mathcal{R}(y) = \sqrt{d}\, y/\|y\|$ and differentiate:

$$\partial\mathcal{R}(x) = \frac{\sqrt{d}}{\|x\|}\left(I - \frac{xx^\top}{\|x\|^2}\right) = \frac{1}{\mathrm{RMS}(x)}\left(I - \hat{x}\hat{x}^\top\right),$$

using $\|x\| = \sqrt{d}\,\mathrm{RMS}(x)$. The matrix $I - \hat{x}\hat{x}^\top$ is the orthogonal projection onto the complement of $\mathrm{span}\{x\}$, with eigenvalues in $\{0, 1\}$, so the operator norm of $\partial\mathcal{R}(x)$ is $1/\mathrm{RMS}(x)$. □

Lemma A.2 transfers the analysis to the sandwich block of Eq. (3.1). At initialization the normalization gains are one, the loop entry has unit RMS by the input-embedding normalization, and every later state is an output of the outer normalization, so $\mathrm{RMS}(\mathbf{x}_i) = 1$ at every visit. By part *(i)* the inner normalization then evaluates the branch, and hence every parameter derivative $\partial f_j/\partial\phi_j$, at the unchanged input $\mathcal{R}(\mathbf{x}_i) = \mathbf{x}_i$; by part *(ii)* each linearized state-propagation path acquires an additional factor of operator norm one at each branch traversal. The visit-wise constants of Assumption 3.1, and the constant C' of Proposition A.3, are therefore unchanged or smaller for the sandwich block. The tied bound of Proposition A.4 is stated through the alignment coefficient κ_R of Eq. (3.7), which is defined for the architecture at hand: because the sandwich block satisfies the same visit-wise bounds $\|U_{r,j}\| \le C_U\beta/\alpha$ and $\|G_{r,j}\| \le C_G\beta/\alpha$, the triangle inequality again gives $0 \le \kappa_R \le R$, and Proposition A.4, Corollary A.5, and Proposition 3.2 hold for Eq. (3.1) exactly as stated, with the worst-case aligned regime $\kappa_R = \Theta(R)$, and hence the $p = 1/2$ exponent, unchanged. The hierarchical reasoners of Section 4 use the plain post-normalized block without inner normalization, for which the propositions apply directly.

Proposition A.3 (Depth-untied first-order stability)**.** For a depth-N post-normalized Transformer whose residual-sublayer parameters are not shared across unrolled depth, with $M = 2N$ residual-sublayer visits, Assumption 3.1 implies the first-order bound Eq. (3.4), and consequently $M(\beta/\alpha)^2 = O(1)$ (Eq. (3.5)) is a sufficient first-order stability condition. This statement does not depend on whether the input and output token embeddings are tied.

Proof of Proposition A.3. In the depth-untied residual stack, every unrolled residual-sublayer visit has its own residual-branch parameter tensor. The first-order change is therefore a sum of M independent visit-wise perturbation terms. By Assumption 3.1, each term is bounded by an output sensitivity $O(\beta/\alpha)$ times an effective update $O(\beta/\alpha)$. Summing over M visits gives

$$\|\Delta F\| \leq C' M(\beta/\alpha)^2,$$

for a depth-independent constant C', which implies the stated sufficient condition. □

Proposition A.4 (Tied loop perturbation)**.** Under Assumption 3.1, a looped Transformer with $J = 2K$ physical residual sublayers and R visits satisfies the first-order bound Eq. (3.8), so $M\kappa_R(\beta/\alpha)^2 = O(1)$ (Eq. (3.9)) is a sufficient tied-depth stability condition.

Corollary A.5 (Worst-case aligned loop bound)**.** If $\kappa_R = \Theta(R)$, then Eq. (3.9) reduces to $MR(\beta/\alpha)^2 = O(1)$.

Proof of Proposition A.4. Using Eq. (3.6) and submultiplicativity,

$$\|\Delta F_{\text{tied}}\| \leq \eta \sum_{j=1}^{J} \left\| \sum_{r=1}^{R} U_{r,j} \right\| \left\| \sum_{t=1}^{R} G_{t,j} \right\| + O(\eta^2).$$

By the definition of κ_R, each physical sublayer contributes at most $R\kappa_R C_U C_G(\beta/\alpha)^2$. Summing over J physical sublayers gives

$$JR\kappa_R C_U C_G(\beta/\alpha)^2 = M\kappa_R C_U C_G(\beta/\alpha)^2.$$

Absorbing η, C_U, and C_G into the constant proves Eq. (3.8). □

Proof of Proposition 3.2. For the scaling family $\alpha = (cN)^p$, $\beta = (dN)^{-p}$, the update-to-residual ratio is

$$\frac{\beta}{\alpha} = (cd)^{-p} N^{-2p}. \tag{A.2}$$

Substituting Eq. (A.2) into the tied-depth condition gives

$$M\kappa_R \left(\frac{\beta}{\alpha}\right)^2 = 2N\,\Theta(R^\gamma)\,(cd)^{-2p} N^{-4p}.$$

At fixed physical depth K, $R = N/K$, so the N-dependence is $\Theta(N^{1+\gamma-4p})$. This quantity remains uniformly bounded as $R \to \infty$ if and only if $1 + \gamma - 4p \leq 0$, equivalently $p \geq (1+\gamma)/4$. □

B Extended derivations and comparisons

B.1 Per-matrix use of the DeepNorm initialization gain

DeepNorm's β in Eq. (2.3) is applied per matrix. If $\mathcal{S}_\ell^{\mathrm{DN}}$ denotes the residual-branch matrices scaled by DeepNet in sublayer ℓ, such as the value and output projections in attention and the feed-forward matrices, then DeepNorm initializes

$$W_{\ell,q}^{(0)} = \beta_{\mathrm{DN}}\, \widetilde{W}_{\ell,q}^{(0)}, \qquad q \in \mathcal{S}_\ell^{\mathrm{DN}}, \qquad \beta_{\mathrm{DN}} = (8N)^{-1/4}, \tag{B.1}$$

with the usual unscaled initializer used for $\widetilde{W}_{\ell,q}^{(0)}$. The quantity that enters the perturbation argument is therefore

$$\frac{\beta_{\mathrm{DN}}}{\alpha_{\mathrm{DN}}} = \frac{(8N)^{-1/4}}{(2N)^{1/4}} = \frac{1}{2\sqrt{N}}. \tag{B.2}$$

B.2 Aligned tied-loop bound for the scaling family

For the scaling family $\alpha = (cN)^p$, $\beta = (dN)^{-p}$ of Proposition 3.2, the worst-case aligned tied-depth quantity of Corollary A.5 evaluates to

$$MR\left(\frac{\beta}{\alpha}\right)^2 = 2NR\,(cd)^{-2p} N^{-4p}. \tag{B.3}$$

At fixed K, $R = N/K$, so Eq. (B.3) is bounded only when $p \geq 1/2$, and DeepNorm's $p = 1/4$ leaves a residual growth of $\Theta(R)$. With the DeepLoop choice $\alpha = (2N)^{1/2}$ and $\beta = (8N)^{-1/2}$,

$$\frac{\beta}{\alpha} = \frac{1}{4N}, \qquad MR\left(\frac{\beta}{\alpha}\right)^2 = 2NR \cdot \frac{1}{16N^2} = \frac{1}{8K},$$

so the worst-case aligned tied-depth bound remains $O(1)$ at fixed physical depth K.

B.3 Side-by-side comparison with DeepNorm

Table 4: Side-by-side comparison of the DeepNorm and DeepLoop scaling rules, summarizing Section 3.

	DeepNorm	DeepLoop
Residual scale α	$(2N)^{1/4}$	$(2N)^{1/2}$
Init gain β	$(8N)^{-1/4}$	$(8N)^{-1/2}$
Per-matrix use of β	$W^{(0)} \leftarrow \beta\,\widetilde{W}^{(0)}$	$W^{(0)} \leftarrow \beta\,\widetilde{W}^{(0)}$
Update-to-residual β/α	$1/(2\sqrt{N})$	$1/(4N)$
Untied bound $M(\beta/\alpha)^2$	$\Theta(1)$	$\Theta(N^{-1})$
Aligned tied-loop bound $MR(\beta/\alpha)^2$ at fixed K	$\Theta(R)$	$\Theta(1/K)$

C HIERARCHICAL RECURRENT REASONERS: EXTENDED ANALYSIS

This appendix records the full setup, bookkeeping, and derivations behind Section 4, together with structural remarks omitted from the main text.

C.1 SETUP AND VISIT BOOKKEEPING

Let $\mathcal{H}$ contain K_H physical blocks (each with two residual sublayers, attention and MLP) and let $\mathcal{L}$ contain K_L physical blocks of the same shape. The hierarchical blocks use the post-normalized recurrence in the form of Eq. (3.1) without the inner branch normalization, which is the form for which the propositions of Appendix A are stated directly, so the analysis applies unchanged. Counting attention and MLP residual sublayers over the C outer cycles, the total unrolled visit count is

$$M = 2\,C\,(K_H + C_L K_L), \tag{C.1}$$

which generalizes the single-module count $M = 2KR$ used in Section 3. The set of distinct physical residual sublayers has size $J = J_H + J_L$ with $J_H = 2K_H$ and $J_L = 2K_L$. Each visit $i = (c, \ell, j)$ has an outer-cycle index c, an inner $\mathcal{L}$-iterate index ℓ (only meaningful for $j \in \mathcal{L}$), and a physical-sublayer index j.

C.2 ONE-STEP GRADIENT APPROXIMATION AND THE PER-MODULE CONDITION

Hierarchical recurrent reasoners are typically trained with a one-step gradient approximation: the forward pass uses all C outer cycles, but the backward pass is computed against the last cycle only, with $\mathbf{z}_H$ and $\mathbf{z}_L$ from earlier cycles detached from the autograd graph. This truncation does not alter the numerical forward values, so Lemma A.1 applies unchanged locally. The perturbation bound of Eq. (4.2) is for the truncated training graph: the states entering the last outer cycle are treated as fixed inputs, and both the sensitivity terms and the update terms from earlier cycles are absent. A perturbation bound for the full untruncated forward map after an optimizer step would require an additional forward-visible read factor.

The per-physical-sublayer round counts on the gradient-visible cycle are

$$R_{\mathrm{g}}^{(\mathcal{H})} = 1, \qquad R_{\mathrm{g}}^{(\mathcal{L})} = C_L. \tag{C.2}$$

Two structural facts simplify Eq. (4.2). First, $\kappa_{\mathrm{g}}^{(\mathcal{H})} \le 1$ trivially: with $R_{\mathrm{g}}^{(\mathcal{H})} = 1$, each physical $\mathcal{H}$-sublayer contributes a single-term inner sum, and the alignment coefficient collapses to a constant. The truncation thus places $\mathcal{H}$ in the untied-DeepNorm regime even though $\mathcal{H}$ is forward-visited C times. Second, $\kappa_{\mathrm{g}}^{(\mathcal{L})} \in [0, C_L]$ by the triangle inequality applied to the inner $\mathcal{L}$-iterate sum. Substituting these facts, upper-bounding $\kappa_{\mathrm{g}}^{(\mathcal{H})}$ by one, and applying the per-module counts in Eq. (C.2) yields the sufficient stability condition of Eq. (4.3), whose gradient-visible aggregate satisfies $\bar{\kappa}_{\mathrm{g}} \in [\,J_H/M_{\mathrm{g}},\, 1 + J_L C_L (C_L - 1)/M_{\mathrm{g}}\,]$.

The decomposition of Eq. (4.2) also licenses a per-module scaling family. Each summand depends only on the (α, β) used at the corresponding module, so a hierarchical reasoner may be parameterized with $(\alpha_{\mathcal{H}}, \beta_{\mathcal{H}})$ and $(\alpha_{\mathcal{L}}, \beta_{\mathcal{L}})$ chosen independently. The admissible exponent at each module is determined by the corresponding summand of Eq. (4.2), not by the global visit count.

C.3 LIMIT CASES FOR THE EXPONENT THRESHOLD

We record the two limit cases of Eq. (4.4) that mirror Proposition 3.2.

Decorrelated inner cycles ($\kappa_{\mathrm{g}}^{(\mathcal{L})} = O(1)$). Then $\bar{\kappa}_{\mathrm{g}} = O(1)$ and Eq. (4.4) is bounded as $M_{\mathrm{g}} \to \infty$ if and only if $p \ge 1/4$. Both modules sit in the untied-DeepNorm regime: the $\mathcal{H}$ module by truncation, and the $\mathcal{L}$ module by the inner-cycle decorrelation.

Aligned inner cycles ($\kappa_{\mathrm{g}}^{(\mathcal{L})} = \Theta(C_L)$). Then the $\mathcal{L}$ contribution to $\bar{\kappa}_{\mathrm{g}}$ is $\Theta(C_L^2\, J_L/M_{\mathrm{g}})$. Two sub-cases follow from how M_{g} is grown. *(i)* If C_L is fixed and $K_H, K_L \to \infty$, then $\bar{\kappa}_{\mathrm{g}} = \Theta(1)$ and the threshold is again $p \ge 1/4$. *(ii)* If K_H, K_L are fixed and $C_L \to \infty$, then $M_{\mathrm{g}} = \Theta(C_L)$ and

$\bar{\kappa}_{\mathrm{g}} = \Theta(C_L)$, so Eq. (4.4) is bounded if and only if $p \geq 1/2$, which recovers the fixed-physical-depth tied-loop threshold of Proposition 3.2 along the inner-cycle axis.

As argued in Section 4, these two limits are a binding constraint and a slack one for the architecture as actually built: the once-per-step $\mathcal{H}$ module is truncated to a single update term and requires only $p \geq 1/4$, while the $\mathcal{L}$ module grows the model's effective depth along the aligned inner-cycle axis of sub-case (ii) and requires $p \geq 1/2$. A single shared residual exponent must satisfy the more demanding module, and $p = 1/2$ simultaneously satisfies the once-per-step module (for which $1/2 \geq 1/4$).

C.4 Forward signal at the loop entry

A practical detail of hierarchical reasoners is the addition of a learned task or puzzle embedding $\mathbf{e}_{\mathrm{task}}$ to the token embedding $\mathbf{e}_{\mathrm{tok}}$ before the loop. The choice of whether to apply RMSNorm to $\mathbf{e}_{\mathrm{tok}} + \mathbf{e}_{\mathrm{task}}$ at the loop entry affects only the forward signal scale $\mathrm{RMS}(\mathbf{x}_0)$ in Lemma A.1; it does not enter Eq. (4.2) or the threshold derivation (4.4). Input-side normalization and the residual scaling rule are therefore separable design choices within this framework, even though both can independently influence the relative magnitude of task-conditioning information across repeated visits.

D MEDIUM-SCALE EXPERIMENTAL DETAIL

This appendix collects the medium-scale material summarized in Sections 5.1 and 5.2: the validation-loss curves (Figure 2), the full per-task downstream table (Table 5), and the per-task downstream figure (Figure 3).

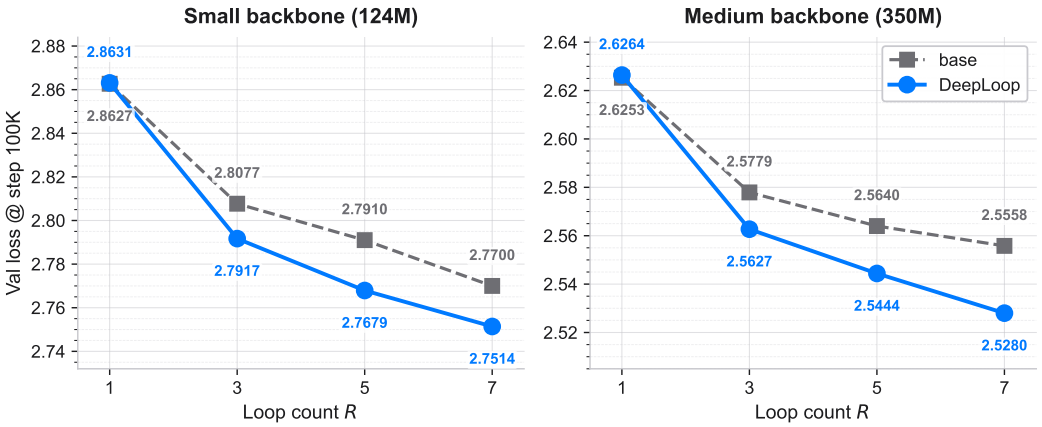


Figure 2: Final validation loss (step 100,000) against loop count $R \in \{1, 3, 5, 7\}$ on FineWeb-Edu 50BT for the GPT-2 small (left) and GPT-2 medium (right) backbones. DeepLoop matches the baseline at R=1 and improves over it at every $R \geq 3$ at both scales in these single-seed runs. Data are the same as Table 1.

Table 5: Downstream accuracy (%) for the GPT-2 medium backbone on FineWeb-Edu 50BT, mirroring Table 6 but extended to R=7. Metric is `acc`; Avg is the unweighted arithmetic mean across the eight tasks. Bold marks the best cell per column per shot setting.

Method / R	ARC-C	ARC-E	HellaSwag	OBQA	PIQA	SciQ	SIQA	WG	**Avg**
0-shot									
baseline R=1	29.44	65.24	38.12	23.60	69.80	85.80	40.23	54.70	50.86
baseline R=3	32.59	66.75	39.76	27.60	70.13	87.30	39.87	55.96	52.50
baseline R=5	**33.53**	68.31	41.01	25.80	69.70	88.20	38.95	55.41	52.61
baseline R=7	32.34	69.40	40.61	24.40	**70.67**	89.20	39.66	57.30	52.95
DeepLoop R=1	29.86	65.07	38.21	23.60	68.82	86.10	39.92	55.25	50.85
DeepLoop R=3	32.59	68.43	40.85	26.80	70.29	88.30	40.02	56.12	52.93
DeepLoop R=5	**33.53**	68.60	41.78	**28.20**	70.46	90.10	40.58	56.12	53.67
DeepLoop R=7	33.11	**69.65**	**41.83**	25.60	70.57	**90.30**	**40.94**	**59.04**	**53.88**
1-shot									
baseline R=1	30.38	66.92	37.73	24.20	70.08	88.30	40.79	55.09	51.69
baseline R=3	33.70	68.69	39.71	26.00	70.13	90.50	40.74	55.33	53.10
baseline R=5	35.75	70.88	40.82	27.80	**71.60**	91.90	39.71	56.91	54.42
baseline R=7	34.22	70.54	40.87	27.80	71.33	92.10	41.30	58.80	54.62
DeepLoop R=1	30.80	65.78	37.73	22.80	69.10	89.50	41.30	55.25	51.53
DeepLoop R=3	33.11	68.14	40.59	26.20	69.48	91.30	41.15	57.38	53.42
DeepLoop R=5	**36.43**	68.18	41.50	26.00	70.84	92.10	40.99	57.46	54.19
DeepLoop R=7	35.15	**70.96**	**41.83**	**28.80**	70.51	**92.40**	**42.53**	**59.43**	**55.20**

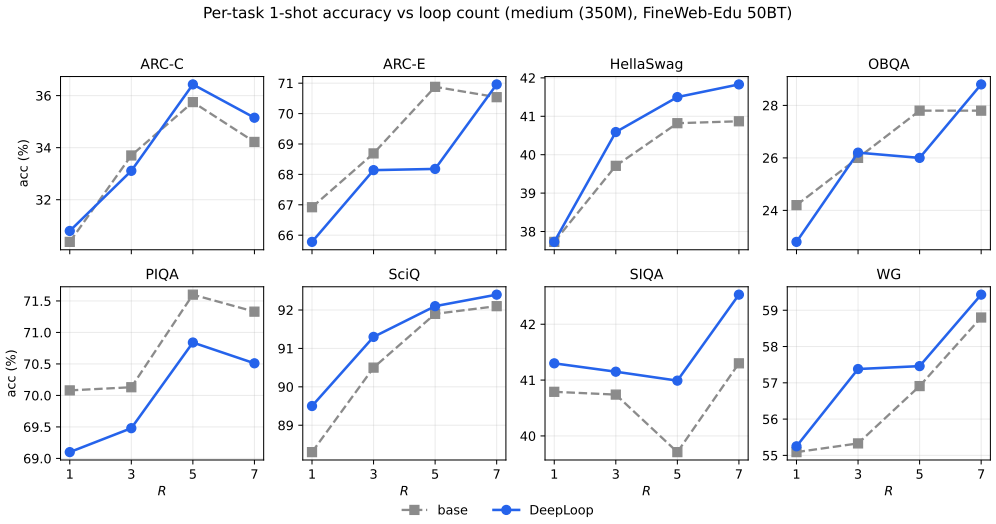


Figure 3: Per-task 1-shot accuracy against loop count R for base versus DeepLoop on all eight tasks at the GPT-2 medium scale. At R=7 DeepLoop wins on seven of the eight tasks under both shot settings; see Table 5 for exact values.

E SMALL-SCALE DOWNSTREAM EVALUATION

We report the GPT-2 small version of the downstream evaluation from §5.2. The architecture is identical to the medium backbone except for hidden size 768 (vs. 1024) and 12 layers (vs. 24); training uses the same FineWeb-Edu 50BT data and schedule but on 4×H200 141 GB GPUs. The same eight-task `lm-evaluation-harness` protocol applies. Table 6 reports per-task accuracy and Avg; Figures 4 and 5 visualize the 1-shot Avg and the per-task 1-shot accuracy, respectively.

Table 6: Downstream accuracy (%) for the GPT-2 small backbone. Metric is `acc`; Avg is the unweighted arithmetic mean across the eight tasks. Bold marks the best cell per column per shot setting.

Method / R	ARC-C	ARC-E	HellaSwag	OBQA	PIQA	SciQ	SIQA	WG	**Avg**
0-shot									
base R=1	23.12	57.11	31.95	20.60	65.83	81.90	37.87	51.62	46.25
base R=3	23.89	60.40	33.70	20.20	66.76	83.20	37.21	52.09	47.18
base R=5	26.62	**61.49**	34.14	20.80	66.49	**83.80**	38.54	53.12	48.12
DeepLoop R=1	24.49	58.38	32.31	20.40	65.83	80.90	38.23	51.30	46.48
DeepLoop R=3	**26.96**	60.31	33.98	**22.00**	67.79	**83.80**	**39.87**	53.20	**48.49**
DeepLoop R=5	25.09	61.36	**34.86**	21.00	**68.50**	82.20	39.46	**53.75**	48.28
1-shot									
base R=1	26.11	57.95	31.95	19.80	65.78	83.50	38.38	52.57	47.00
base R=3	24.74	61.28	33.42	20.80	**67.68**	86.80	37.87	52.41	48.13
base R=5	26.88	62.08	33.92	**23.20**	67.52	**87.10**	**40.02**	52.57	49.16
DeepLoop R=1	25.26	57.15	31.90	21.00	67.14	82.80	38.89	51.30	46.93
DeepLoop R=3	26.88	61.41	33.91	21.00	67.46	**87.10**	39.76	53.83	48.92
DeepLoop R=5	**28.24**	**62.67**	**34.64**	22.60	67.41	87.00	**40.02**	**54.70**	**49.66**

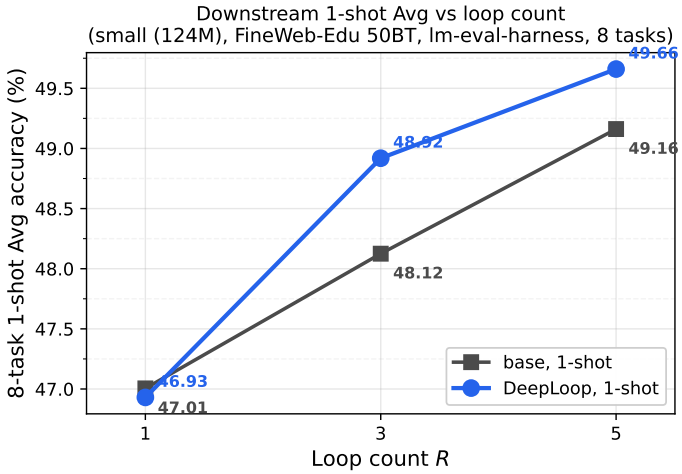


Figure 4: Eight-task 1-shot Avg accuracy against loop count R for the GPT-2 small backbone, base versus DeepLoop. Numbers match the 1-shot Avg column of Table 6.

The two methods are within ±0.25 points of each other at R=1 on both shot settings, and DeepLoop has higher Avg at R=3 (by +1.31 and +0.79 points on 0-shot and 1-shot respectively) and at R=5 (by +0.15 and +0.50). The headline small-scale cell is DeepLoop R=5 1-shot at 49.66 %. Against the same-R baseline (base R=5 1-shot, 49.16 %), DeepLoop wins on four of the eight tasks (ARC-C, ARC-E, HellaSwag, WinoGrande), is within 0.11 points on three tasks (PIQA, SciQ, SIQA), and loses clearly only on OpenBookQA.

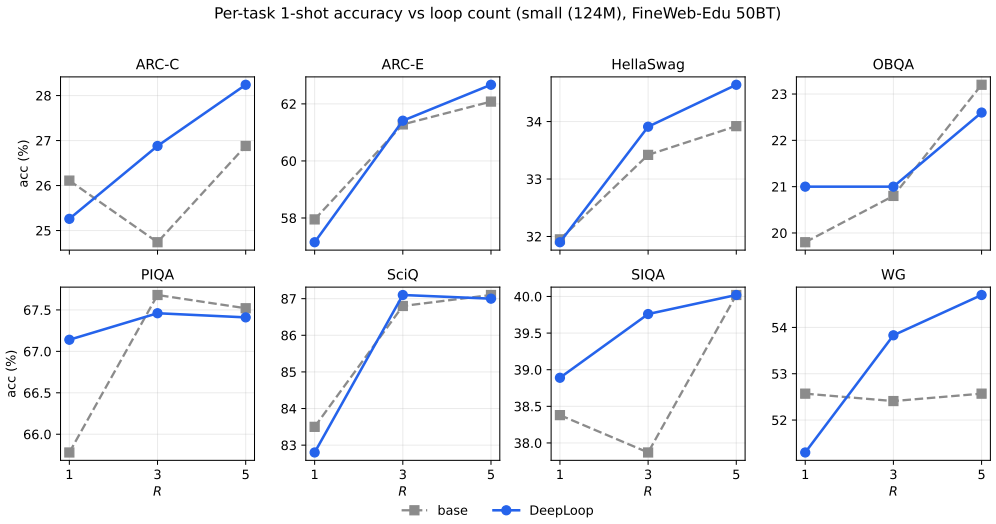


Figure 5: Per-task 1-shot accuracy against loop count R for base versus DeepLoop on all eight tasks at the GPT-2 small scale. Numbers match Table 6.

F EMPIRICAL p-SWEEP AT FIXED LOOP COUNT

Proposition 3.2 predicts a $p{=}1/2$ training-stability threshold under the worst-case aligned regime. We test this prediction directly with a single-axis sweep over the exponent p at fixed $R{=}3$ on the GPT-2 small backbone. Every sweep cell uses the sandwich block of Eq. (3.1). All other ingredients, tied embeddings, input-embedding RMSNorm, FineWeb-Edu 50BT data pipeline, batch size, learning rate, and seed protocol, are identical to the DeepLoop runs in Table 1; only the residual scaling $\alpha = (2N)^p$ and initialization gain $\beta = (8N)^{-p}$ change. We run a grid of seven values, $p \in \{0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60\}$, with up to five seeds per value. Each cell is a 90-minute run on $4\times$H200 141 GB GPUs, which under our throughput reaches roughly 2,700 optimizer steps; we report validation loss at the common comparison checkpoint of step 2000 where available, and use the trailing-50-iter mean train loss as a surrogate for runs that do not reach a post-warmup validation step.

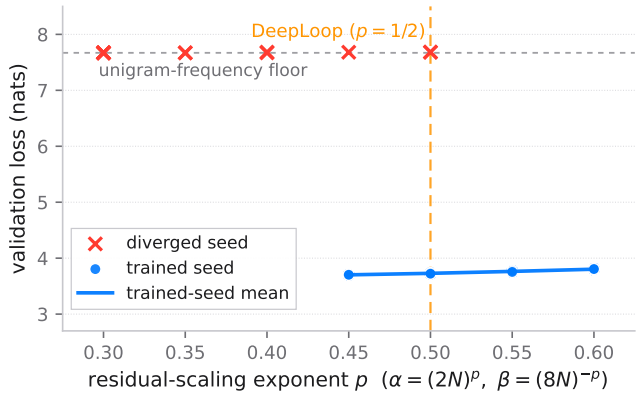


Figure 6: Validation-loss snapshot at step 2000 on FineWeb-Edu 50BT for the GPT-2 small backbone, $R{=}3$, sweeping the residual-scaling exponent $p \in \{0.30, \ldots, 0.60\}$. Each marker is the most recent attempt for one (p, seed) cell; several cells were relaunched, and the pooled per-seed escape counts over all attempts are reported in the text. Red crosses are attempts that do not escape the unigram-frequency floor (≈ 7.67 nats); blue circles are attempts that train; the blue line connects the per-p trained-attempt mean. Escape becomes more frequent as p increases, and every seed trains at $p \geq 0.55$; the transition brackets the Proposition 3.2 worst-case prediction of $p{=}1/2$. Conditional on convergence, larger p gives a slightly higher loss (less aggressive learning).

The picture in Figure 6 is consistent with the analysis. Several (p, seed) cells were launched more than once for scheduling reasons; we count a seed as escaping if *any* of its attempts reaches a validation evaluation below the unigram floor (≈ 7.67 nats) within the budget. Under this pooled count, the per-seed escape fractions are $0/5$, $1/5$, $2/5$, $2/5$, $3/5$, $5/5$, and $5/5$ for $p = 0.30, \ldots, 0.60$: escape is the minority outcome in the clearly sub-threshold region $p \leq 0.40$, remains unreliable through $p = 0.50$, and becomes universal from $p = 0.55$. At this short horizon the transition is gradual rather than sharp, and it brackets the $p{=}1/2$ worst-case prediction of Proposition 3.2 rather than pinpointing it. Second, among the plotted attempts that do train, the mean validation loss at the same step rises monotonically with p (3.70 at $p{=}0.45$, 3.73 at $p{=}0.50$, 3.76 at $p{=}0.55$, 3.80 at $p{=}0.60$), confirming that more aggressive scaling (smaller p) delivers a stronger learning signal whenever it does not destabilize training. We adopt $p{=}1/2$ as the DeepLoop default on the strength of the worst-case analysis; the sweep places the empirical transition around it, with failures concentrated below it and the loss penalty for exceeding it growing with p. Choices $p < 1/2$ produce lower trained-attempt mean loss when they do train, but with at least $3/5$ of seeds failing to escape the floor at every $p \leq 0.45$, they are not safe defaults at this horizon.

The p-sweep is small in scope: it is at one scale (GPT-2 small), one loop depth ($R{=}3$), and one optimizer-step budget. Whether the same boundary at $p{=}1/2$ applies at larger K, at different normalization placements, or at substantially longer training, is left to future work.

G COMPUTE RESOURCES

All experiments were run on NVIDIA H200 141 GB GPUs in a SLURM cluster. GPT-2 small cells use $4\times$H200, and GPT-2 medium cells use $8\times$H200. Per-cell wall-clock time is read from the corresponding WandB run records; aggregate GPU-hours below are rounded to the nearest hundred.

Per-cell cost. Wall-clock time scales roughly linearly with the loop count R:

	R=1	R=3	R=5	R=7
Small wall-clock (h)	9	22	35	50
Small GPU-hours / cell	37	90	145	200
Medium wall-clock (h)	12	32	52	77
Medium GPU-hours / cell	95	260	420	620

Aggregate cost. The reported FineWeb-Edu language-modeling matrix (Tables 1 and 5: 16 cells $\{\text{base, DeepLoop}\} \times \{\text{small, medium}\} \times R \in \{1, 3, 5, 7\}$) totals approximately $\mathbf{3{,}700}$ H200 GPU-hours. Exploratory experiments not included in the main paper, a range of optimization-related variants and additional benchmark sweeps we evaluated during the development of the method and ultimately did not report, add approximately $\mathbf{7{,}000}$ GPU-hours.

The total project compute is therefore approximately $\mathbf{10{,}700}$ H200 GPU-hours.