BEYOND THE LOOP: FROM LATENT DYNAMICS TO PERFORMANT RECURRENT MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recurrent Transformers offer a parameter-efficient way to increase latent computation through repeated application of shared modules. However, it remains unclear how to design better-performing recurrent architectures, train them effectively and efficiently, and assess how broadly their gains transfer. To address these questions, we examine HRM-Text as a representative structured recurrent model and find that repeated L-module calls refine the state toward an input-conditioned basin, while the H-module applies a distinct transformation. This motivates $L^n H$, a simpler single-stream recurrent family that preserves this functional separation. Under the original training recipe, $L^7 H$ matches HRM-Text at equal trainable and executed depth; with a WSD50–Muon recipe, its eight-benchmark mean improves by 4.8–5.7 points at both 0.69B and 1.0B and surpasses recipe-matched HRM-Text. We further show that truncated backward windows preserve useful gradient directions, while shorter differentiable paths and parameter sharing reduce 60B-token training time by more than 50% relative to matched-depth standard Transformers in our $8\times$H100 production setting. We finally examine how these gains extend beyond the training mixture. They persist on source-matched evaluations and controlled arithmetic shifts, while broader transfer remains mixed under the current breadth and scale of pretraining. As a complementary adaptability test, fine-tuning HRM-Text on a safety domain absent from pretraining substantially improves safety performance while preserving most of its general-task utility.

1 INTRODUCTION

Transformer language models conventionally obtain more computation by stacking independently parameterized layers (Vaswani et al., 2017). Recurrent Transformers offer a different scaling axis: by reusing modules across depth, they can increase latent computation without a proportional increase in trainable parameters (Dehghani et al., 2019; Geiping et al., 2025; Zhu et al., 2025). Recent recurrent language models instantiate this idea in different ways. Huginn (Geiping et al., 2025) places a shared recurrent Transformer core between distinct prelude and coda modules. RINS (Alabdulmohsin & Zhai, 2025) repeats an earlier block before a later transformation, and HRM-Text (Wang et al., 2026) couples low-level and high-level modules through a staged recurrent computation. However, recurrent depth is qualitatively different from simply adding more independently parameterized layers: the same transformation is repeatedly applied to hidden states produced by earlier applications of that transformation. It is therefore unclear what kinds of computation emerge from recurrence, how reliably such computation can be trained as recurrent depth grows, and whether its benefits persist beyond the settings in which it is optimized. This leaves three key questions largely open: (i) what computation should be repeated, (ii) how should recurrent depth be trained effectively and efficiently, and (iii) how well do recurrent models generalize beyond their training data?

We address these questions through HRM-Text (Wang et al., 2026), which provides a useful testbed for studying them together. It alternates repeated updates of a low-level L-module with less frequent updates of a high-level H-module, maintains separate latent states across two stages, and trains the full recurrent forward computation with a truncated backward window. Its strong reasoning performance under a comparatively modest pretraining budget raises a natural question: which parts of this machinery are responsible for the gain? To separate these effects, we combine matched architectural comparisons with analyses of recurrent dynamics, training behavior, and generalization.

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.

Our analysis first asks what HRM-Text's hierarchy is doing that simpler forms of recurrence do not. Its advantage cannot be explained by recurrence, parameter sharing, or depth alone. Instead, inference-time interventions and latent-state analyses reveal a functional asymmetry: repeated L-module calls refine the latent state toward an input-conditioned basin, a narrow region that trajectories from different initial states converge to, while the H-module performs a distinct transformation that moves the state away from that region. This separation motivates $L^n H$, which removes HRM-Text's two-stream, two-stage hierarchy but keeps both roles. Its structure follows the $A^r B$ organization of RINS (Alabdulmohsin & Zhai, 2025). Relative to the RINS baseline in HRM-Text's implementation, which injects x_0 into every repeated call, $L^n H$ also injects x_0 into the terminal H call and uses HRM-Text's truncated-gradient curriculum. At equal depth, $L^7 H$ matches HRM-Text under the original recipe.

Even with a simpler recurrent architecture, training remains a second challenge: parameter sharing reduces model size, but full backpropagation still spans the executed trajectory. HRM-Text instead truncates the backward graph while retaining the full forward computation. We show that these truncated gradients remain directionally informative relative to exact finite-depth backpropagation, consistent with a local equilibrium-motivated analysis. Separately, the optimization recipe matters: a WSD50–Muon recipe yields substantial gains for $L^7 H$ at both model scales and keeps it ahead of recipe-matched HRM-Text. In our 8×H100 setting, the shorter differentiable graphs cut 60B-token training time by more than 50% relative to standard Transformers at equal executed depth.

Recurrent gains matter only if they extend beyond the training distribution. We therefore separate in-distribution (source-matched) from out-of-distribution evaluation. The gains persist on source-matched held-out data and controlled arithmetic shifts, while broader transfer is more mixed and generally trails models pretrained on substantially larger and broader corpora. In addition, safety fine-tuning of HRM-Text on an unseen domain induces the target behavior while largely preserving general-task utility. Figure 6 in Appendix A overviews our main contributions:

(1) Mechanism and architecture. We show that HRM-Text's repeated L-module calls contract different initial states into an input-conditioned basin, while its H-module applies a distinct transformation. Guided by this, the single-stream $L^n H$ instance $L^7 H$ matches HRM-Text at equal depth.

(2) Training recurrent computation. We show that truncated backward windows stay positively aligned with exact gradients, consistent with the contraction in (1). Combined with parameter sharing, truncation more than halves training time relative to matched-depth standard Transformers.

(3) Improved models. We will release $L^n H$ models trained with our WSD50–Muon recipe on the same 60B-token data as HRM-Text; at 1.0B, the recipe raises $L^7 H$'s eight-benchmark mean from 73.9 to 78.7. We also show that these gains persist on source-matched data and controlled arithmetic shifts, while broader transfer remains mixed, plausibly limited by the breadth of pretraining.

2 Recurrent Transformers and HRM-Text

2.1 Recurrent Computation in Transformers

Let $x_0 \in \mathbb{R}^{T\times d}$ denote the embedded token-sequence representation. A standard Transformer obtains depth from independently parameterized modules (Vaswani et al., 2017), whereas a standard looped Transformer repeatedly applies one shared module (Dehghani et al., 2019; Giannou et al., 2023):

$$\begin{aligned} &\text{Standard Transformer:} && h_0 = x_0, && h_k = F_{\theta_k}(h_{k-1}), && k = 1,\dots,N, \\ &\text{Standard looped Transformer:} && h_0 = h_{\text{init}}, && h_k = F_{\theta}(h_{k-1} + \alpha x_0), && k = 1,\dots,N. \end{aligned}$$

For the standard Transformer, $\theta_1,\dots,\theta_N$ are distinct. In the standard looped Transformer, h_{init} initializes the recurrent state and $\alpha \in \{0,1\}$ toggles input injection of x_0. A shared d_F-layer module repeated N times has trainable depth d_F but executed depth Nd_F. At each autoregressive decoding step, all recurrent calls precede the next-token logits, adding latent computation without emitting intermediate tokens. Appendix B reviews related recurrent-depth models in greater detail.

2.2 HRM-Text Architecture

HRM (Wang et al., 2025) separates a fast-updating low-level module (the L-module) from a slower high-level module (the H-module). HRM-Text (Wang et al., 2026) adapts this hierarchy to 60B-token pretraining at 0.69B and 1.0B, each trained in about 50 hours on 8 and 16 H100s, respectively.

Architecture. Let z_H and z_L denote the high-level and low-level states. HRM-Text initializes z_H from the token representation x_0, initializes z_L from a fixed random vector broadcast across the sequence, and executes two stages of three L-module calls followed by one H-module call:

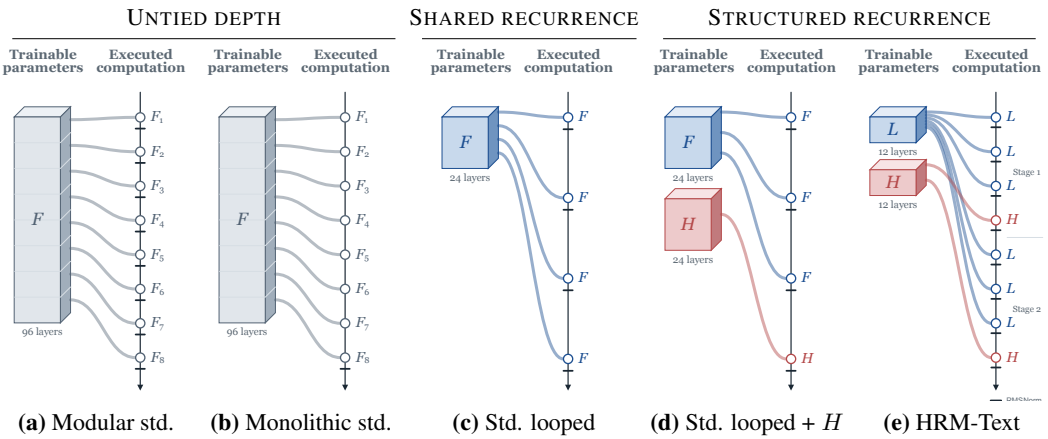
$$\begin{aligned} z_H^0 &= x_0, \qquad z_L^0 = z_{L,\text{init}}, \\ z_L^i &= L_{\theta_L}(z_L^{i-1} + z_H^0),\ i \in \{1,2,3\}, \quad z_H^1 = H_{\theta_H}(z_H^0 + z_L^3), \\ z_L^i &= L_{\theta_L}(z_L^{i-1} + z_H^1),\ i \in \{4,5,6\}, \quad z_H^2 = H_{\theta_H}(z_H^1 + z_L^6). \end{aligned} \tag{1}$$

We abbreviate the schedule as $3L \to H \to 3L \to H$. Every L-module call receives the current z_H, each H-module call receives the latest z_L, and z_H^2 feeds the LM head. Training uses instruction–response pairs, response-only next-token loss, and a PrefixLM mask (Dong et al., 2019) (bidirectional over the instruction, causal over the response). All eight calls execute in the forward pass. During the first 20% of training, the backward window expands from the last two calls to the last five; afterwards only these receive gradients, leaving the first three L-module calls forward-only.

2.3 Recurrence and Depth Alone Do Not Explain HRM-Text's Gain

Before analyzing HRM-Text's internals, we ask whether simpler explanations account for its advantage. HRM-Text couples two latent states, distinct shared L- and H-modules, and a two-stage $3L \to H \to 3L \to H$ schedule. We use the 0.69B HRM-Text configuration as the reference and construct four controls to isolate plausible sources of its advantage. Let D_f denote executed depth and D_p trainable depth; all five models use $D_f = 96$. The modular and monolithic standard Transformers both use $D_p = 96$ and test whether HRM-Text's gain can be explained by full unshared depth. They differ only in whether RMSNorm separates eight 12-layer segments, mirroring the module boundaries of the recurrent models; standard Transformers do not normally use such boundary normalization. The standard looped Transformer reuses one 24-layer F-module four times ($D_p = 24$), isolating parameter sharing and recurrence without hierarchy. The standard looped Transformer with a distinct H-module replaces the final repeated call with a separately parameterized 24-layer module ($D_p = 48$), testing a distinct final transformation without HRM-Text's two latent streams and staged conditioning, but with a larger parameter budget. HRM-Text retains the complete hierarchy with shared 12-layer L- and H-modules ($D_p = 24$). All controls share the data, objective, and training budget; backward windows and warmup differ (Appendix Table 6; computation graphs in Figure 1).

Table 1 evaluates ARC-Challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), DROP (Dua et al., 2019), GSM8K (Cobbe et al., 2021), HellaSwag (Zellers et al., 2019), MATH (Hendrycks et al., 2021b), MMLU (Hendrycks et al., 2021a), and WinoGrande (Sakaguchi et al., 2020). The comparison yields three observations. *(i)* HRM-Text outperforms the standard looped Transformer despite matching it in parameter count, trainable depth, and executed depth, so recurrence and sharing alone do not explain its gain. *(ii)* It also outperforms both 96-layer standard controls on average,



(a) Modular std. **(b)** Monolithic std. **(c)** Std. looped **(d)** Std. looped + H **(e)** HRM-Text

Figure 1: Architecture controls at executed depth $D_f = 96$. Each curve links a set of parameters to every call that uses it (reuse, not data flow). Repeated symbols share parameters, whereas F_i denotes a distinct module; crossbars mark a final RMSNorm at each module boundary.

showing that unshared depth is insufficient. *(iii)* Replacing the fourth F-module call with a distinct H-module transformation improves performance consistently. Together, these observations motivate the architectural questions that follow: what distinct roles do the L- and H-modules play, how does recurrent depth affect performance, and which elements of the two-stream hierarchy are necessary?

Table 1: Performance at executed depth 96. D_p, D_f: trainable and executed depth; (a)–(e) as in Figure 1. DROP is token-level F1, the rest accuracy (%); Mean is the unweighted average.

Model (params)	D_p	D_f	ARC	BoolQ	DROP	GSM8K	Hella.	MATH	MMLU	Wino.	Mean
(a) Modular std., 2.28B	96	96	62.54	81.44	65.57	67.40	36.15	43.36	49.50	59.35	58.16
(b) Monolithic std., 2.28B	96	96	**75.94**	83.88	73.72	74.30	42.87	49.58	56.02	64.88	65.15
(c) Std. looped, 0.69B	24	96	66.21	75.84	69.95	67.55	40.06	43.56	50.57	61.96	59.46
(d) Std. looped + H, 1.22B	48	96	73.38	82.48	76.00	77.10	47.29	50.34	51.12	64.17	65.23
(e) HRM-Text, 0.69B	24	96	75.68	**84.50**	**78.69**	**79.45**	**52.00**	**51.34**	**56.76**	**67.32**	**68.22**

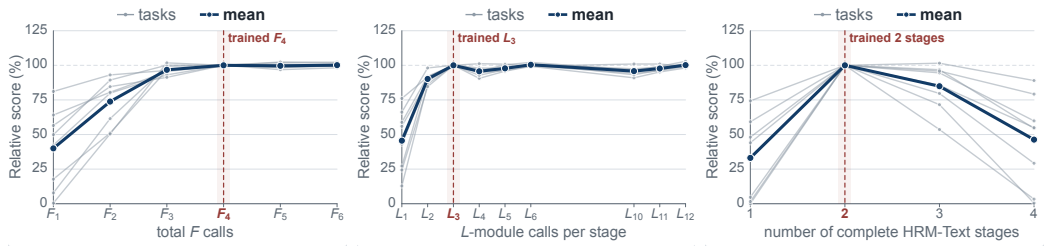
3 What HRM-Text's Modules Compute

3.1 Depth Interventions Reveal a Staging Asymmetry

To test whether HRM-Text's L- and H-modules play different computational roles, we intervene on the trained recurrent schedule at inference time, varying recurrent depth while holding the weights fixed. As a reference, we first sweep the standard looped Transformer: its mean score rises through the trained four-call depth and remains essentially unchanged at five and six calls (Figure 2a). Repeating a shared transformation can produce fixed-point or attractor dynamics, as studied in deep equilibrium models (Bai et al., 2019) and recurrent reasoning models (Huang et al., 2026a; Fein-Ashley & Rashidinejad, 2026; Yang et al., 2026); the observed saturation fits this behavior.

HRM-Text instead exhibits a staging asymmetry: its two modules respond differently to extra calls. Its repeated L-module calls mirror the standard looped Transformer structurally and show a similar saturation pattern: fewer than three calls per stage hurt performance, whereas additional calls keep it within a few points of the trained setting (Figure 2b). Individual benchmarks dip by up to about five points between multiples of the trained three-call stage length and recover at those multiples (Appendix Table 8). By contrast, removing or adding a whole $3L \to H$ stage clearly hurts (Figure 2c).

> **Takeaway:** Extra L-module calls cost at most about five points per benchmark, whereas removing a complete $3L \to H$ stage collapses most benchmarks and adding one sharply hurts the generative ones (GSM8K 79.5 to 56.9); the L-module thus acts as a loop, and the H-module does not.



(a) Std. looped: F-module calls. **(b)** HRM-Text: L-module calls. **(c)** HRM-Text: complete stages.

Figure 2: Full-evaluation inference-depth sweeps with fixed checkpoints, normalized per task to the trained setting (100). Gray curves show tasks, dark blue their mean, and dashed red the trained depth.

3.2 The L-Module Contracts Toward an Input-Conditioned Basin

Trained-Path L-Module Contraction. To understand why additional L-module calls cost little, we follow the hidden state through the default forward pass, starting from $z_{L,\text{init}}$, and measure the size of each L-module step with $d_{\cos}(u,v) = 1 - \cos(u,v)$ and $d_{\text{rel}}(u,v) = \|v-u\|_2/\|u\|_2$. Both step sizes shrink within each stage (Table 2a). After the first H-module call, H_1, changes the conditioning state from z_H^0 to z_H^1, Stage 2 shows the same shrinking steps, consistent with convergence.

Table 2: State distances on the 200-example probe set (smaller is closer). (a) Consecutive L-module updates shrink within both trained stages. (b) Each H-module output stays far from the continued L-module state, and continuation endpoints differ across stages.

(a) Trained-path contraction

Stage	Transition	$d_{\cos}$	d_{rel}
1	$z_{L,\text{init}} \to z_L^1$	0.9301	1.3415
	$z_L^1 \to z_L^2$	0.4749	0.9737
	$z_L^2 \to z_L^3$	0.2443	0.6909
2	$z_L^3 \to z_L^4$	0.2443	0.6930
	$z_L^4 \to z_L^5$	0.1232	0.4932
	$z_L^5 \to z_L^6$	0.0732	0.3788

(b) Distinct H-module transformation

Comparison	$d_{\cos}$	d_{rel}
z_H^1 vs. $\tilde{z}_{L,0}^{(10)}$	0.6690	1.1560
z_H^1 vs. $\tilde{z}_{L,0}^{(100)}$	0.6690	1.1560
z_H^2 vs. $\tilde{z}_{L,1}^{(10)}$	0.6676	1.1552
z_H^2 vs. $\tilde{z}_{L,1}^{(100)}$	0.6693	1.1566
$\tilde{z}_{L,0}^{(10)}$ vs. $\tilde{z}_{L,1}^{(10)}$	0.1856	0.6064
$\tilde{z}_{L,0}^{(100)}$ vs. $\tilde{z}_{L,1}^{(100)}$	0.1890	0.6120

Cross-Initialization Contraction. Smaller successive updates along a single trajectory provide only partial evidence of a basin. We say the L-module has an input-conditioned basin if trajectories from different low-level initial states, under the same conditioning state z_H, contract toward a common region; this is closely related to path independence in learned dynamical systems (Anil et al., 2022). We therefore draw 256 random initial z_L states with the norm of the default initialization, holding $z_H^0 = x_0$ fixed, and repeat this at the Stage 2 input under fixed z_H^1. For states $\{u_i\}_{i=1}^M$, we report:

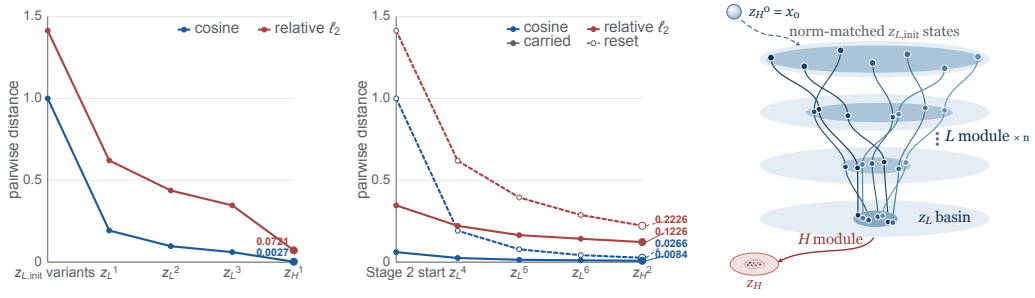
$$D_{\cos}(\{u_i\}) = \binom{M}{2}^{-1} \sum_{i<j} \left(1 - \frac{u_i^\top u_j}{\|u_i\|_2 \|u_j\|_2}\right), \quad D_{\text{rel}}(\{u_i\}) = \left(\frac{\binom{M}{2}^{-1} \sum_{i<j} \|u_i - u_j\|_2^2}{M^{-1} \sum_i \|u_i\|_2^2}\right)^{1/2}.$$

With $z_H^0 = x_0$ fixed, Figure 3 shows contraction in both stages: the Stage 1 pairwise cosine distance falls from 1.0001 to 0.0611 after three L-module calls, and Stage 2 states contract whether they continue from the Stage 1 endpoints ("carried") or restart from fresh random states ("reset"). The final H_2 states from all 256 Stage 1 starts also land near the default one (Appendix Table 13). Over 100 additional calls, local motion remains limited (step size 0.0160) and the downstream H_1 output remains nearly invariant (cosine distance 0.0057; Appendix Table 9).

Takeaway: Repeated L-module calls pull all 256 sampled starting states into a narrow, input-conditioned basin that stays narrow over 100 extra calls.

3.3 The H-Module Performs a Distinct Transformation

If the H-module merely continued the refinement, its output would lie close to the state reached by further L-module calls under the same conditioning. We call it a distinct transformation if instead it moves the state far from that basin. Let $\tilde{z}_{L,r}^{(m)}$ denote the state after m additional L-module calls under fixed z_H^r. Table 2b shows that both H-module outputs remain 0.6690 and 0.6693 in cosine



(a) Stage 1 controlled starts. **(b)** Stage 2 under fixed z_H^1. **(c)** Recurrent-basin schematic.

Figure 3: Contraction from norm-matched low-level-state interventions. (a) Pairwise spread (cosine, relative ℓ_2) of 256 Stage 1 starts. (b) "Carried" propagates each Stage 1 endpoint z_L^3, whereas "reset" uses a fresh norm-matched state; both continue under fixed z_H^1. (c) Schematic: repeated L-module calls enter an input-conditioned basin, which the H-module maps to a distinct high-level state.

distance from the corresponding L-module states even after 100 continuation calls, far more than the 0.0732 of the last L-module step. The two stages also reach different basins (distance 0.1890), so each H-module output re-conditions the next stage. Extra H-module calls eventually make only small state updates, yet move away from the trained logits and hurt performance, supporting a distinct transformation rather than recurrent refinement (Appendix Table 10).

Takeaway: The H-module output lies far from further L-module refinement, and repeating it hurts.

Together, the experiments support a basin-then-transformation account, $y = F_{\text{out}}(\mathcal{B}_L(x_0); x_0)$: repeated L-module calls contract the state into an input-conditioned basin $\mathcal{B}_L(x_0)$, and a separate transformation F_{out} that does not refine under iteration maps this basin to the output.

4 A Single-Stream Recurrence, L^nH, Matches HRM-Text

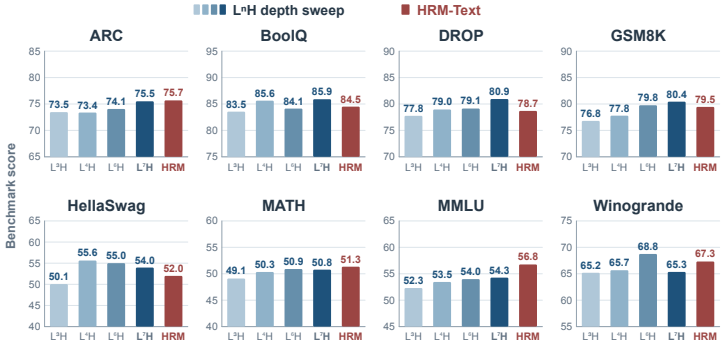
If the only roles are refinement by the L-module and a separate transformation by the H-module, HRM-Text's second stream and its intermediate H-module update should be unnecessary. We test this with a single-stream recurrence that keeps both roles, which we call L^nH:

$$\begin{aligned} s_0 &= s_{\text{init}}, \qquad s_k = L(s_{k-1} + x_0)\,, \quad k = 1, \ldots, n, \\ y &= H(s_n + x_0)\,. \end{aligned} \tag{2}$$

Here s_{init} uses the same fixed random initialization scheme as HRM-Text's low-level state, and the L- and H-modules are independently parameterized Transformers with d_L and d_H layers. Unlike the standard looped Transformer, L^nH shares the L-module across its n recurrent calls but keeps the final H-module separate, so L can specialize in refinement and H in mapping the refined state to a prediction. Its structure follows the A^rB organization of RINS (Alabdulmohsin & Zhai, 2025), with $A = L$, $B = H$, and $r = n$; it is the minimal schedule implied by the analysis above, not a new building block. Unlike the RINS baseline in HRM-Text's implementation, which injects x_0 only into the repeated calls, L^nH also injects x_0 into the terminal H-module call and uses HRM-Text's truncated-gradient curriculum. Appendix C.5 shows that terminal injection adds 1.66 points (66.74 to 68.40); also removing repeated injection lowers it further but changes the gradient route. Executing the shared L-module n times gives $D_f(L^nH) = nd_L + d_H$, while sharing keeps $D_p(L^nH) = d_L + d_H$; with $d_L = d_H = 12$, L^7H matches HRM-Text's 96-layer executed and 24-layer trainable depth. With a single stream, L^7H preserves the motivating mechanism: Appendix Table 16 shows contraction and a distinct H-module transformation like HRM-Text's.

Selecting Recurrent Depth. Contraction alone does not say how many recurrent calls are needed, so we train L^3H, L^4H, L^6H, and L^7H at 0.69B under one recipe, with executed depths of 48, 60, 84, and 96 layers. The mean improves gradually with depth (Figure 4), and shorter models lose little: L^3H reaches 66.03 with half the executed depth. L^6H matches HRM-Text's 68.22, and L^7H reaches 68.40, slightly above HRM-Text, at the same 96-layer executed depth.

Takeaway: A single stream suffices: L^7H matches HRM-Text at equal depth without its hierarchy.



Model	Exec.	Hours	Mean
L^3H	48	36.8	66.03
L^4H	60	44.7	67.61
L^6H	84	49.7	68.22
L^7H	96	52.5	**68.40**
HRM-Text	96	52.3	68.22

Figure 4: Benchmark scores with increasing recurrent L-module depth for 0.69B models. The side table reports executed depth, approximate training time, and the unweighted eight-benchmark mean.

5 TRAINING RECURRENT TRANSFORMERS

Recurrent depth adds forward computation without proportional parameter growth, yet full backpropagation still spans the entire executed trajectory. HRM-Text reduces this cost with truncated backpropagation (Williams & Peng, 1990; Tallec & Ollivier, 2017): after warmup, all eight module calls run forward, but only the last five receive gradients (Appendix Figure 8). Is this truncated gradient a reliable training signal? We compare it with exact backpropagation and with an equilibrium-based reference, then study how the training recipe and systems choices affect quality and cost.

5.1 TRUNCATED GRADIENTS THROUGH RECURRENT DEPTH

Exact Finite Comparison. Let K count the module calls that receive gradients. We compare the training gradient ($K = 5$) with the exact one through all eight calls ($K = 8$). With $K = 5$, the first three L-module calls are detached; the H-module and LM-head gradients remain exact, while shared L-module and embedding gradients keep only their paths through the last five calls. Input injection gives every retained call of L^7H and the looped Transformer, but not of HRM-Text, a direct route from the embeddings to the loss. Over ten batches after each of epochs 1–3, the native truncated gradient (5 of 8 calls; 2 of 4 for the looped Transformer) stays positively aligned with the exact gradient (Table 3).

Equilibrium-Motivated Explanation. The contraction in Section 3.2 suggests why a short tail remains informative. Consider how a perturbation of the Stage 2 conditioning state z_H^1 propagates through the retained L-module calls to the input of H_2. Under the $K = 5$ window this gives the finite transport $T_2^{(5)}$, which we compare with the local equilibrium response T_2^{IFT} given by the implicit function theorem (IFT). Let $J^\star = J_{L,2}^{\mathrm{in},\star}$ denote the local L-module input Jacobian at the Stage 2 three-call state; the Neumann expansion of $(I - J^\star)^{-1}$ converges when $\rho(J^\star) < 1$:

$$T_2^{(5)} = I + J_{L,6}^{\mathrm{in}} + J_{L,6}^{\mathrm{in}} J_{L,5}^{\mathrm{in}} + J_{L,6}^{\mathrm{in}} J_{L,5}^{\mathrm{in}} J_{L,4}^{\mathrm{in}}, \qquad T_2^{\mathrm{IFT}} = (I - J^\star)^{-1} = I + J^\star + (J^\star)^2 + (J^\star)^3 + \cdots .$$

When late-call Jacobians vary slowly in the contracted region, the retained paths correspond to the first terms of the equilibrium response. Appendix D.1 derives this. Treating the observed three-call states as approximate local equilibria, we build an IFT-style gradient reference; the truncated gradients also align with it (0.70–0.95 for HRM-Text and L^7H, 0.45–0.59 for the looped Transformer; Table 3).

Takeaway: The truncated gradient stays positively aligned with the exact gradient, consistent with the contracting L-module making the retained calls approximate the equilibrium response.

Table 3: Cosine similarity of the truncated training gradient with the exact gradient and the IFT-style reference over epochs 1–3 (ten batches each). Calls: retained/total. Full results: Appendix D.2.

Architecture	Parameter group	Calls	vs. full finite	vs. IFT-style
HRM-Text	L-module	$5/8$	0.824–0.877	0.703–0.832
	Embeddings	$5/8$	0.513–0.653	0.804–0.815
L^7H	L-module	$5/8$	0.699–0.952	0.736–0.918
	Embeddings	$5/8$	0.848–0.960	0.861–0.954
Std. looped	Shared Transformer block	$2/4$	0.833–0.930	0.450–0.537
	Embeddings	$2/4$	0.746–0.928	0.479–0.589

5.2 AN IMPROVED TRAINING RECIPE

Final quality also depends on how the retained gradient is optimized. HRM-Text's original recipe uses AdamATan2 (Everett et al., 2024) with 2,000 warmup steps followed by a constant learning rate. Holding the architecture and backward window fixed, we combine the WSD50 schedule (Hu et al., 2024; Wen et al., 2025; Hägele et al., 2024), which decays the learning rate over the second half of training, with Muon (Jordan et al., 2024; Liu et al., 2025) for Transformer weight matrices, implemented through an adapted `muon_fsdp_2` path (Jaghouar, 2025). At 0.69B, this WSD50–Muon recipe raises the eight-benchmark mean from 68.22 to 72.57 for HRM-Text and from 68.40 to 74.06 for L^7H (Appendix Figure 9), while increasing training time by about 12% (Appendix

Table 32). Across both architectures and scales, the recipe improves nearly every benchmark and the mean. The 1.0B L^7H model is also competitive with published models with larger parameter or training budgets (Table 4), though most benchmarks are source-matched (Section 6).

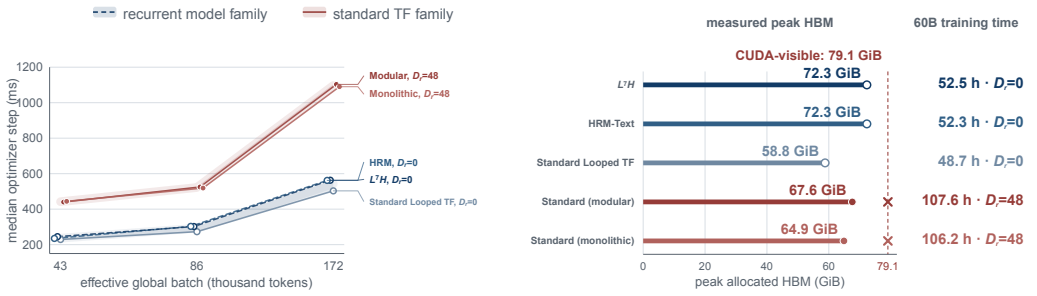
Takeaway: WSD50–Muon improves both models by 3.5–5.7 points; L^7H stays better on average.

Table 4: Comparison with published models; seven of eight benchmarks are source-matched.

Model	Arch.	Tokens (T)	MMLU	ARC-C	Hella.	Wino.	BoolQ	DROP	GSM8K	MATH
Ours										
L^7H 1B, WSD50–Muon	Recurrent	0.06	62.3	**91.6**	76.1	74.2	**90.1**	**85.8**	**88.1**	**61.0**
HRM-Text 1B, WSD50–Muon	Recurrent	0.06	59.7	90.2	73.0	**74.7**	89.6	82.2	86.6	57.5
Fully open, reported by HRM-Text										
HRM-Text 1B (Wang et al., 2026)	Recurrent	0.06	60.7	81.9	63.4	72.4	86.2	82.2	84.5	56.2
Huginn 3.5B (Geiping et al., 2025)	Recurrent	0.8	31.4	38.2	65.2	59.4	69.8	17.8	34.6	12.6
Olmo 3 7B (Team Olmo, 2025)	Dense	6	65.8	81.6	72.7	64.6	85.4	71.5	75.5	40.0
Open weight, reported by HRM-Text										
Llama 3.2 3B (Grattafiori et al., 2024)	Dense	9	58.0	69.1	47.1	52.4	76.2	45.2	77.7	48.0
Gemma 3 4B (Gemma Team, 2025)	Dense	4	59.6	56.2	**77.2**	64.7	72.3	60.1	38.4	24.2
Qwen3.5 2B (Qwen Team, 2026)	Dense	36	64.5	81.0	64.6	56.7	80.5	30.8	53.0	34.2
Ouro 1.4B (Zhu et al., 2025)	Recurrent	7	**67.4**	60.9	74.3	72.3	83.6	49.7	78.9	22.4

5.3 Sharing and Truncation Halve Training Time

Truncation also shortens the differentiable graph. At equal executed depth $D_f = 96$, we compare all models by trainable depth D_p, retained-gradient depth D_g (layers that receive gradients), and recomputation depth D_r (layers recomputed by activation checkpointing; Chen et al., 2016). These measures separate forward computation and parameter count from the depth that must remain differentiable or be recomputed during training. Timings for HRM-Text and L^7H include their $K = 2 \to 5$ warmup; Appendix Table 32 gives the full configurations.



(a) Optimizer-step time across batch sizes.

(b) Production memory and training time.

Figure 5: Production efficiency at $D_f = 96$ on eight H100 GPUs; lower is better. (a) Median step time vs. global batch, annotated with D_r at the largest. (b) Peak GPU memory (HBM) and 60B-token time at the 172,032-token batch; crosses at 79.1 GiB mark OOM runs with $D_r = 0$.

Across batch sizes, the three recurrent models form a similar lower-latency band, whereas both standard Transformers take roughly twice as long at the 172,032-token production batch (Figure 5). There, the recurrent models fit in memory with $D_r = 0$, whereas the standard Transformers need $D_r = 48$. The speedup comes from the shorter retained-gradient depth and, at this batch, from avoiding that recomputation. Sharing has only a modest direct effect at matched D_g and D_r but matters indirectly: because every layer is reused in the last calls, the backward pass can stop early without leaving any layer untrained, and the shorter pass needs no recomputation. Appendix F separates these effects.

Takeaway: Parameter sharing plus truncation avoids activation recomputation and more than halves 60B-token training time (52.5 vs. 106.2 hours) at equal executed depth in our 8×H100 setting.

6 How Far Do the Gains Generalize?

Training Mixture and In-Distribution Generalization. Our models are trained on a stratified instruction–response mixture rather than broad raw-text pretraining, spanning seven source groups and 40B unique tokens (60B processed positions); Appendix Table 33 details the datasets and benchmark overlap. Seven of the eight core benchmarks are source-matched to this mixture and evaluated on held-out splits. We therefore treat these results primarily as source-matched in-distribution (ID) generalization, and reserve out-of-distribution (OOD) for controlled shifts and external benchmarks. One source-matched symbolic source warrants closer inspection: Sudoku-Extreme. Appendix G.3 shows that models trained on it copy its format and clues but rarely solve it, with little aggregate benefit.

> **Takeaway:** On the seven held-out evaluations of Appendix Table 34, L^7H leads in every scale–recipe setting, by 0.2–0.4 points in three and 1.7 at 1.0B with WSD50–Muon (83.6 vs. 81.9).

Out-of-Distribution Generalization. We next evaluate two levels of transfer beyond source-matched data: controlled mathematical shifts with GSM-Symbolic (Mirzadeh et al., 2025), and broader knowledge, reasoning, and instruction transfer with MMLU-Pro (Wang et al., 2024), BBH (Suzgun et al., 2023), and IFEval (Zhou et al., 2023); see Table 5 (score sources in Appendix G.2).

Table 5: Transfer beyond source-matched data: GSM8K and three GSM-Symbolic variants (controlled) and knowledge, reasoning, and instruction following (broader). IFEval is run locally at 0 and 3 shots (Gemma 3 4B IT: official 0-shot score); "–": no comparable 3-shot protocol.

		Controlled mathematical transfer				Broader transfer				
Model	Recipe	GSM8K	GSM-Sym. main	GSM-Sym. p_1	GSM-Sym. p_2	MMLU	MMLU-Pro	BBH	IFEval 0-shot	IFEval 3-shot
HRM-Text, 1.0B	AdamATan2	84.4	83.5	68.5	46.4	59.8	**33.2**	**41.7**	45.1	42.9
HRM-Text, 1.0B	WSD50–Muon	86.6	82.3	70.3	47.1	59.7	30.0	35.1	44.5	43.0
L^7H, 1.0B	AdamATan2	85.2	79.7	68.5	47.4	59.5	28.2	41.3	**49.6**	**48.0**
L^7H, 1.0B	WSD50–Muon	**88.1**	**85.3**	**75.9**	**58.7**	**62.3**	30.6	39.8	41.4	40.4
Olmo 3 7B Base (Team Olmo, 2025)	–	75.5	68.4	51.8	25.3	66.5	37.2	63.5	**55.4**	**62.1**
Gemma 3 4B Base (Gemma Team, 2025)	–	40.9	37.1	18.8	4.1	59.6	29.2	50.9	34.8	51.6
Qwen3 8B Base (Yang et al., 2025)	–	**89.8**	**84.3**	**76.3**	**58.7**	**76.9**	**56.7**	**78.4**	48.6	52.2
Ouro 1.4B Base (Zhu et al., 2025)	–	78.5	76.3	62.3	39.8	67.5	48.6	71.0	40.2	55.3
Olmo 3 7B Instruct-SFT (Team Olmo, 2025)	–	90.0	85.2	75.3	59.2	66.9	45.7	51.0	86.2	–
Gemma 3 4B IT (Gemma Team, 2025)	–	87.4	83.8	73.9	52.0	58.1	43.6	72.2	**90.2**	–
Olmo 3 7B Think-SFT (Team Olmo, 2025)	–	**94.7**	**92.2**	**90.3**	**89.2**	**76.6**	**62.6**	**84.1**	86.9	–

Across increasingly difficult GSM-Symbolic shifts, the 1.0B L^7H under WSD50–Muon has a growing margin over recipe-matched HRM-Text, showing that its gains extend beyond source-matched evaluation to controlled arithmetic transfer. On broader benchmarks, performance is less consistent and generally trails models trained on substantially larger and broader corpora, consistent with limitations from the breadth and scale of our pretraining. As a complementary domain-transfer test, we fine-tune HRM-Text on safety data absent from pretraining; the model acquires the target behavior while preserving most general-task utility (Appendix G.4). These results suggest that broader pretraining coverage and targeted domain adaptation are promising directions beyond the current mixture.

> **Takeaway:** Under WSD50–Muon, L^7H's margin over HRM-Text grows with shift difficulty (58.7 vs. 47.1 on GSM-Sym. p_2), but broader transfer is mixed, and the recipe lowers IFEval and BBH.

7 Conclusion

Latent-state interventions in HRM-Text reveal recurrent L-module refinement followed by a distinct H-module transformation, motivating L^nH, which removes the two-stream, two-stage hierarchy while matching HRM-Text at equal depth. WSD50–Muon improves performance across scales and recurrent architectures, keeping L^7H ahead of recipe-matched HRM-Text. Truncated backward windows preserve useful gradient directions and, with sharing, substantially reduce training cost relative to matched-depth standard Transformers. Gains extend to source-matched data and controlled arithmetic shifts, while broader transfer remains mixed. Together, these results provide practical guidance for organizing recurrent computation, training it efficiently, and evaluating where its gains transfer.

REPRODUCIBILITY STATEMENT

Our experiments build on the public HRM-Text codebase and data pipeline (Wang et al., 2026). Model definitions and recurrent schedules are given in Sections 2 and 4, Table 6 (0.69B), and Appendix E (1.0B); the mechanism probes, probe sets, and intervention protocols are in Section 3 and Appendix C. The truncated-gradient, implicit-function (IFT), and Neumann-series derivations are in Appendix D. Optimizer settings, including the WSD50 schedule, warmup, learning rates, Muon's parameter assignment, and the AdamATan2 configuration, are in Appendix E; hardware, batch sizes, and systems profiles are in Section 5.3 and Appendix F. The training mixture and its benchmark overlap are in Appendix G.1, evaluation protocols for all compared models in Section 6 and Appendix G.2, and the safety fine-tuning and HarmBench settings in Appendix G.4.

ETHICS STATEMENT

This work involves no human subjects, crowdsourcing, or new data collection. All training and evaluation data are publicly released datasets, listed with their sources in Appendix G.1 (training), Section 6 and Appendix G (evaluation), and Appendix G.4 (safety fine-tuning); some are web-derived, and we rely on their curators' filtering rather than auditing them for personal or harmful content ourselves. The safety experiments fine-tune on public red-teaming datasets that contain harmful requests, use them only to train refusal behavior, and report only aggregate HarmBench scores; we did not measure over-refusal. Like other language models, the $L^n H$ models we plan to release can produce incorrect or harmful text and are intended for research on recurrent architectures, not for deployment without further safety evaluation and fine-tuning.

AI USE STATEMENT

We used generative AI tools (large language models) to assist with software implementation and debugging, language editing for clarity and readability, and improving the presentation of mathematical derivations written by the authors. AI tools did not produce any reported experimental result, and the authors decided which experiments to run and which claims to make. All AI-assisted code, text, and mathematical derivations were reviewed and verified by the authors. The authors take full responsibility for the final content of the paper, including any remaining errors.

REFERENCES

Pranjal Aggarwal, Marjan Ghazvininejad, Seungone Kim, Ilia Kulikov, Jack Lanchantin, Xian Li, et al. Reasoning over mathematical objects: On-policy reward modeling and test time aggregation, 2026. URL https://arxiv.org/abs/2603.18886.

Ibrahim Alabdulmohsin and Xiaohua Zhai. Recursive inference scaling: A winning path to scalable inference in language and multimodal systems. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/9cf64a372841aee60c059923e3289048-Abstract-Conference.html.

Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J. Zico Kolter, and Roger Grosse. Path independent equilibrium models can better exploit test-time computation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 7796–7809, 2022. doi: 10.52202/068431-0566. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/331c41353b053683e17f7c88a797701d-Abstract-Conference.html.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/01386bd6d8e091c2ab4c7c7de644d37b-Abstract.html.

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. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL

https://proceedings.neurips.cc/paper_files/paper/2022/hash/7f70331dbe58ad59d83941dfa7d975aa-Abstract-Conference.html.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016. URL https://arxiv.org/abs/1604.06174.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. AceReason-Nemotron: Advancing math and code reasoning through reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/9fc2aa7f8a66eed231007b8bee4485cb-Abstract-Conference.html.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal Transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/c20bb2d9a50d5ac1f713f8b34d9aac5a-Abstract.html.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, 2019. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246/.

Katie E. Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12666–12700. PMLR, 2024. URL https://proceedings.mlr.press/v235/everett24a.html.

Run-Ze Fan, Zengzhi Wang, and Pengfei Liu. MegaScience: Pushing the frontiers of post-training datasets for science reasoning. In *Conference on Language Modeling*, 2026a. URL https://arxiv.org/abs/2507.16812.

Ying Fan, Anej Svete, and Kangwook Lee. Bridging the gap between latent and explicit reasoning with looped transformers, 2026b. URL https://arxiv.org/abs/2606.31779.

Jacob Fein-Ashley and Paria Rashidinejad. Solve the loop: Attractor models for language and reasoning, 2026. URL https://arxiv.org/abs/2605.12466.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, et al. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f9e1e8b56c7e363985ebeb0e9dd1a85c-Abstract-Conference.html.

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. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Abstract-Conference.html.

Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442, 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. OLMES: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5020–5048, 2025. doi: 10.18653/v1/2025.findings-naacl.282. URL https://aclanthology.org/2025.findings-naacl.282/.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, et al. OpenThoughts: Data recipes for reasoning models. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/b010241b9f1cdfc7d4c392db899cef86-Abstract-Conference.html.

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://papers.nips.cc/paper_files/paper/2024/hash/8b970e15a89bf5d12542810df8eae8fc-Abstract-Conference.html.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. WildGuard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0f69b4b96a46f284b726fbd70f74fb3b-Paper-Datasets_and_Benchmarks_Track.pdf. Datasets and Benchmarks Track.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021b. URL https://datasets-benchmarks-proceedings.neurips

.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2404.06395.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026a. URL https://openreview.net/forum?id=lh95PnOlpM.

Benhao Huang, Chufan Shi, Junlin Chen, Shicheng Wen, Zhengzhong Liu, Eric Xing, and Xuezhe Ma. Towards Looped Models Done Right—Part I: Topology, Input Injection, Recurrent-State Design. https://ifm-research.notion.site/Towards-Looped-Models-Done-Right-3ade511912ec8128987dfeb7a5580043, 2026b. Accessed 2026-08-31.

Sami Jaghouar. Muon FSDP2: A Muon optimizer compatible with FSDP2. https://github.com/samsja/muon_fsdp_2, 2025. Version 0.3.0.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/54024fca0cef9911be36319e622cde38-Abstract-Conference.html.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/, 2024.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, et al. Tülu 3: Pushing frontiers in open language model post-training. In *Conference on Language Modeling*, 2025. URL https://openreview.net/pdf?id=i1uGbfHHpH.

Ariel N. Lee, Cole J. Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of LLMs. NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following, 2023. URL https://arxiv.org/abs/2308.07317.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath. Hugging Face dataset repository, 2024. URL https://huggingface.co/datasets/AI-MO/NuminaMath-CoT.

Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, KiJung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard Zemel. Reviving and improving recurrent back-propagation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3082–3091, 2018. URL https://proceedings.mlr.press/v80/liao18c.html.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, et al. Muon is scalable for LLM training, 2025. URL https://arxiv.org/abs/2502.16982.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. The Flan collection: Designing data and methods for effective instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22631–22648, 2023. URL https://proceedings.mlr.press/v202/longpre23a.html.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/11332b6b6cf4485b84afadb1352d3a9a-Abstract-Conference.html.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-Reasoner: Advancing LLM reasoning across all domains. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.nips.cc/paper_files/paper/2025/file/51c52cbb4b3faa891ca0621d13efe1be-Paper-Conference.pdf.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35181–35224, 2024. URL https://proceedings.mlr.press/v235/mazeika24a.html.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. In *Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=PXVQTHYwgt.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Iman Mirzadeh, Keivan Alizadeh-Vahid, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ec2e7a896f8250986b3907f57621ce94-Abstract-Conference.html.

Fernando J. Pineda. Generalization of back propagation to recurrent and higher order neural networks. In *Neural Information Processing Systems*, 1987. URL https://papers.nips.cc/paper_files/paper/1987/hash/6ad10c3a760cfad3e5d1fa1ddaefdec2-Abstract.html.

PleIAs. SYNTH. Hugging Face dataset repository, 2025. URL https://huggingface.co/datasets/PleIAs/SYNTH.

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

Qwen Team. Qwen3.5-2B model card. Hugging Face model repository, 2026. URL https://huggingface.co/Qwen/Qwen3.5-2B.

Nazneen Rajani, Lewis Tunstall, Edward Beeching, Nathan Lambert, Alexander M. Rush, and Thomas Wolf. No robots. Hugging Face dataset repository, 2023. URL https://huggingface.co/datasets/HuggingFaceH4/no_robots.

Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986. doi: 10.1137/0907058.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/2676109d49d1eb26d6bc584a8f556305-Abstract-Conference.html.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1gR5iR5FX.

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. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/3501672ebc68a5524629080e3ef60aef-Abstract.html.

Damien Sileo. tasksource: A large collection of NLP tasks with a structured dataset preprocessing framework. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pp. 15655–15684, 2024. URL https://aclanthology.org/2024.lrec-main.1361/.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Corentin Tallec and Yann Ollivier. Unbiasing truncated backpropagation through time, 2017. URL https://arxiv.org/abs/1705.08209.

Team Olmo. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. OpenMathInstruct-2: Accelerating AI for math with massive open-source instruction data. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/302ce0673c00aee2cf84bb43d0117553-Abstract-Conference.html.

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. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025. URL https://arxiv.org/abs/2506.21734.

Guan Wang, Changling Liu, Chenyu Wang, Cai Zhou, Yuhao Sun, Yifei Wu, Shuai Zhen, Luca Scimeca, and Yasin Abbasi Yadkori. HRM-Text: Efficient pretraining beyond scaling, 2026. URL https://arxiv.org/abs/2605.20613.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract.html. Datasets and Benchmarks Track.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape view. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/6a1fe80a9e2dcda0b3e5fd0fd87eb097-Abstract-Conference.html.

Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501, 1990. doi: 10.1162/neco.1990.2.4.490.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026. URL https://openreview.net/forum?id=tbdSfBMlbq.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. ReClor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgJtT4tvB.

Weizhe Yuan, Jane Yu, Song Jiang, Karthik Padthe, Yang Li, Dong Wang, et al. NaturalReasoning: Reasoning in the wild with 2.8m challenging questions. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/60b22138917015fcba7165670e6f453c-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, et al. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12):3848–3860, 2023. doi: 10.14778/3611540.3611569. URL https://www.vldb.org/pvldb/vol16/p3848-huang.pdf.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

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

A Paper Overview

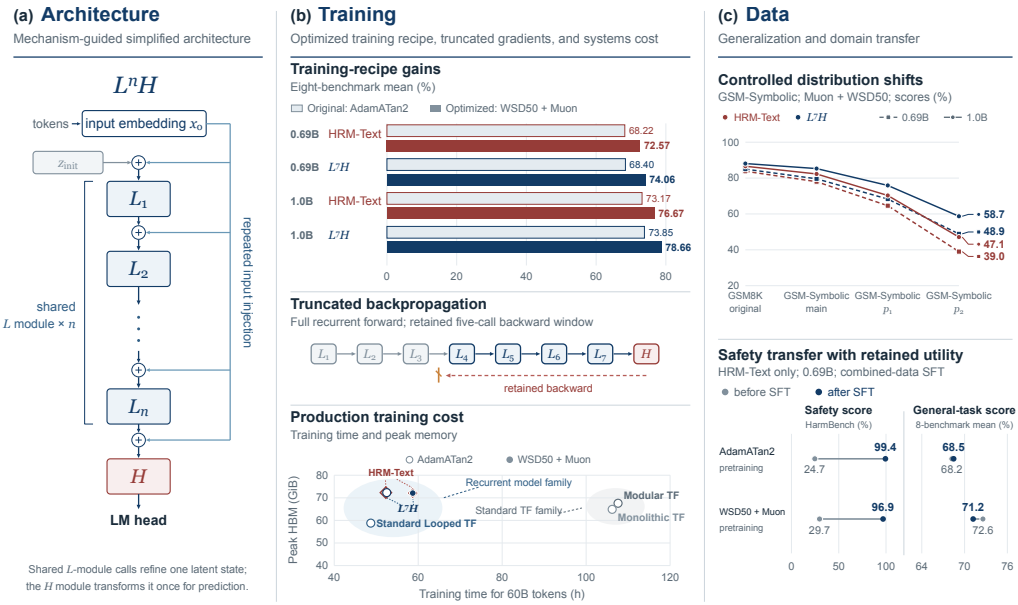


Figure 6: Overview of architecture, training, and domain transfer. (a) L^nH repeatedly refines an input-conditioned state with a shared L-module before one H-module transformation. (b) compares training recipes, depicts five-call backpropagation in an eight-call recurrence, and reports 60B-token training time versus peak HBM. (c) GSM-Symbolic tests controlled shifts; HRM-Text safety SFT evaluates domain transfer while tracking general-task utility.

Table 6: Configurations of the executed-depth-96 controls in Figure 1, plus L^7H. Module depths are Transformer layers; "Backward calls" counts module calls that receive gradients.

Model	Schedule	$L/F, H$ depth	Added conditioning to recurrent calls	Added conditioning to H	Backward calls	Warmup
Modular std. (a)	$F_1 \cdots F_8$	$12, –$	–	–	8 of 8	No
Monolithic std. (b)	F	$96, –$	–	–	1 of 1	No
Std. looped (c)	F^4	$24, –$	x_0 each F	–	2 of 4	No
Std. looped + H (d)	F^3H	$24, 24$	x_0 each F	x_0	2 of 4	No
HRM-Text (e)	$(L^3H)^2$	$12, 12$	current z_H	current z_H	$2 \to 5$ of 8	Yes
L^7H	L^7H	$12, 12$	x_0 each L	x_0	$2 \to 5$ of 8	Yes

For HRM-Text, $z_H^0 = x_0$, while the second H call receives the updated z_H^1, not a fresh direct injection of x_0. Warmup expands the retained window over the first 20% of training.

B Related Work

B.1 Recurrent Depth and Iterative Computation

This work is related to the literature on using parameter sharing across depth to increase executed computation without proportionally increasing model size. Universal Transformers (Dehghani et al., 2019) repeatedly apply a shared self-attention transition, while later work studies how recurrent or looped networks can learn or implement iterative algorithms (Schwarzschild et al., 2021; Bansal et al., 2022; Giannou et al., 2023) and exploit additional latent computation (Saunshi et al., 2025). Related equilibrium and attractor-based models study the dynamics induced by repeated transformations, including fixed points (Bai et al., 2019), path independence (Anil et al., 2022), and learned attractors

(Huang et al., 2026a; Fein-Ashley & Rashidinejad, 2026; Yang et al., 2026). Our models are finite-depth recurrent Transformers rather than implicit equilibrium models, but these works motivate our use of contraction and continuation behavior to characterize recurrent refinement.

Our work contributes to this literature by asking how recurrent computation should be organized. Rather than only varying the amount of shared depth, we compare pure shared recurrence, recurrence followed by a distinct transformation, and HRM-Text's staged hierarchy under matched forward computation and matched training conditions (data, objective, and budget).

B.2 Recurrent Language Models

Recent language models instantiate recurrent depth in different ways. Retrofitted Recurrence (McLeish et al., 2026) and LOTUS (Fan et al., 2026b) add latent computation to pretrained models. Other work trains recurrent architectures end to end. Huginn (Geiping et al., 2025) places a shared recurrent Transformer core between distinct prelude and coda modules, so additional test-time iterations increase latent computation without emitting intermediate reasoning tokens. While Huginn emphasizes test-time depth, Ouro (Zhu et al., 2025) pretrains looped language models with an entropy-regularized objective that learns how to allocate recurrent depth. Increasing the number of loops also raises stability concerns: Parcae (Prairie et al., 2026) constrains injection parameters to mitigate residual explosion and loss spikes, then examines how quality scales with training and test-time computation. Complementing these systems, the controlled study of Huang et al. (2026b) shows that recurrence topology, input injection, and recurrent-state design can materially affect performance; our study complements it by relating such choices to the latent dynamics of trained models and to the truncated gradients used to train them in practice.

B.3 Structured Recurrence: RINS and HRM-Text

The closest architectural precedents are RINS and HRM-Text. RINS repeats an earlier block before applying a later block, yielding an $A^r B$ computation (Alabdulmohsin & Zhai, 2025). Our $L^n H$ recurrence follows the same abstract organization, but arises from examining the distinct roles of repeated and terminal computation in HRM-Text. We therefore view $L^n H$ as the minimal schedule implied by our mechanism analysis rather than as a new building block. The RINS baseline in HRM-Text's public implementation already injects the input into every repeated call, but not into its terminal block, and it is trained with a fixed backward window rather than HRM-Text's warmup. In HRM-Text's own comparison this baseline trails HRM-Text (Wang et al., 2026), whereas $L^7 H$, which adds terminal input injection and the warmup, matches it. We hypothesize that these two differences account for the gap. Appendix C.5 isolates terminal injection ($+1.66$ points at fixed warmup); isolating the warmup at fixed module depth, call count, data, and optimizer is future work.

HRM (Wang et al., 2025) introduces fast low-level and slower high-level modules, and HRM-Text (Wang et al., 2026) adapts this hierarchy to conditional language modeling with two latent streams, two recurrent stages, repeated cross-level conditioning, and a truncated backward window. HRM-Text is both our empirical starting point and our closest controlled baseline. We ask whether its functional separation between repeated L-module updates and the H-module transformation requires the full two-stream, two-stage hierarchy, or whether a single stream suffices.

C Additional Mechanism Analyses and Results

The main text identifies recurrent refinement followed by a distinct state transformation along the trained HRM-Text and $L^7 H$ trajectories. This appendix first reports the values underlying the main-text mechanism results and extended depth interventions, then provides complementary interventions on the same 200-example mechanism slice. These interventions test two consequences of the proposed account. First, a recurrent refinement module should recover from displaced states. Second, a terminal transformation should depend on receiving a refined basin state and need not improve when iterated beyond its trained depth. We test these predictions within HRM-Text (Appendices C.1 and C.2), isolate the refinement behavior in the pure standard looped Transformer (Appendix C.3), and then recompose the two roles in $L^7 H$ (Appendix C.4), and ablate its input injection (Appendix C.5).

C.1 Main-Text Values and Extended Depth Interventions

Unless noted otherwise, the mechanism probes use the final (epoch-4) checkpoint and 200 examples, with 25 each from MMLU, ARC, HellaSwag, Winogrande, BoolQ, GSM8K, MATH, and DROP. Distances flatten token and hidden dimensions before aggregation.

Figure 2 normalizes each benchmark to the trained depth so that the depth trends can be compared directly. Tables 7 and 8 preserve the underlying per-benchmark metrics.

Table 7: Full-evaluation standard looped TF sweep in Figure 2a. The trained depth is F_4. Values are percentages: token-level F1 for DROP, accuracy otherwise, as in Table 1.

Depth	GSM8K	MATH	DROP	MMLU	ARC	Hella.	Wino.	BoolQ
F_1	0.3	3.4	12.4	28.6	28.2	25.7	50.3	37.9
F_2	34.1	26.8	35.7	42.8	53.2	32.3	57.7	67.9
F_3	68.8	43.3	67.5	46.2	62.0	38.3	59.7	**76.1**
$\mathbf{F_4}$	67.6	**43.6**	69.9	**50.6**	**66.6**	40.1	62.0	76.0
F_5	**69.2**	42.9	**71.3**	49.0	65.5	**40.4**	61.4	75.6
F_6	69.1	43.4	71.0	49.7	66.5	40.1	**62.2**	75.4

Table 8: Full-evaluation loop sweeps for the HRM-Text checkpoint in Figure 2. L_k denotes k L-module calls per stage. Values (%): token-level F1 for DROP, accuracy otherwise, as in Table 1.

Depth	GSM8K	MATH	DROP	MMLU	ARC	Hella.	Wino.	BoolQ
Two stages: vary L-module calls per stage								
L_1	10.2	12.5	21.4	33.4	32.6	29.1	51.2	55.9
L_2	68.7	43.4	71.7	52.3	67.2	46.1	61.6	82.9
$\mathbf{L_3}$	79.5	51.3	**78.7**	56.8	75.7	**52.0**	67.3	84.5
L_4	76.1	47.6	74.5	54.2	73.5	47.1	66.0	**85.5**
L_5	79.8	49.7	75.7	55.4	73.5	49.8	65.8	85.2
L_6	81.2	**51.8**	78.4	**57.2**	76.3	51.6	**67.5**	84.0
L_{10}	76.7	47.8	74.2	54.6	73.8	47.3	65.2	85.3
L_{11}	78.9	49.5	75.6	55.5	74.7	49.5	65.4	85.3
L_{12}	**81.8**	**51.8**	78.0	57.1	**76.5**	51.0	66.9	83.8
Vary complete HRM-Text stages								
1 stage	0.0	0.8	3.7	25.0	25.0	25.0	50.0	50.0
2 stages	**79.5**	**51.3**	**78.7**	**56.8**	**75.7**	**52.0**	**67.3**	84.5
3 stages	56.9	27.5	62.7	53.7	73.3	44.5	64.5	**85.8**
4 stages	0.4	1.7	23.0	34.0	41.2	28.6	53.3	75.2

Takeaway: The looped Transformer saturates after its four trained calls. HRM-Text tolerates extra L-module calls per stage (within about five points, peaking at multiples of three) but drops sharply on the generative benchmarks and HellaSwag when a stage is added (GSM8K 79.5 to 56.9, MATH 51.3 to 27.5) and collapses to chance on multiple-choice tasks when one is removed (MMLU 25.0).

Longer L-Module Continuation. To test whether the basin persists beyond the trained endpoint, we hold z_H^0 fixed and continue Stage 1 from $\widetilde{z}_L^{(0)} = z_L^3$ using $\widetilde{z}_L^{(k+1)} = L(\widetilde{z}_L^{(k)} + z_H^0)$. Local motion and cross-initialization spread remain small, cumulative drift saturates, and the downstream H_1 output is nearly unchanged even after 100 additional calls.

Table 9: Longer L-module continuation and downstream H_1 stability. Here k counts additional L-module calls beyond the trained L_3 endpoint. Panel (a) reports continued state dynamics; panel (b) measures the distance from the trained $H_1(z_L^3)$ output.

(a) Dynamics within the recurrent basin

Diagnostic	Comparison	$k=1$	$k=3$	$k=10$	$k=50$	$k=100$
One-step residual	$d_{\cos}(\tilde{z}_L^{(k-1)}, \tilde{z}_L^{(k)})$	0.0997	0.0371	0.0124	0.0156	0.0160
Cross-init spread	$D_{\cos}(\{\tilde{z}_L^{(k,i)}\}_{i=1}^N)$	0.0351	0.0202	0.0149	0.0179	0.0181
Cumulative drift from L_3	$d_{\cos}(\tilde{z}_L^{(0)}, \tilde{z}_L^{(k)})$	0.0997	0.1243	0.1301	0.1332	0.1335

(b) Stability of the downstream H_1 transformation

Stopping point and resulting state	$d_{\cos}$	d_{rel}
2 calls early, $H_1(z_L^1)$	0.0344	0.2564
1 call early, $H_1(z_L^2)$	0.0092	0.1336
1 extra call, $H_1(\tilde{z}_L^{(1)})$	0.0045	0.0913
10 extra calls, $H_1(\tilde{z}_L^{(10)})$	0.0056	0.1032
100 extra calls, $H_1(\tilde{z}_L^{(100)})$	0.0057	0.1044

Takeaway: The Stage 1 basin persists beyond the trained endpoint: after 100 extra L-module calls, the step size is 0.016, drift saturates at 0.13, and the H_1 output moves by only 0.006.

Repeated H-Module Calls. To isolate the H-module, we hold z_L^6 fixed and apply further H-module calls, $z_H^t = H(z_H^{t-1} + z_L^6)$, beyond the trained H_2 endpoint. Though updates shrink, logits drift and the mean score declines, so low motion alone does not make extra H-module calls task-useful.

Table 10: Output drift on the 200-example mechanism slice under additional H-module calls after the trained H_2 endpoint, with z_L^6 held fixed. The state residual compares each H_t with H_{t-1}; logit distance and the changed top-1 fraction use H_2 as the reference. Mean score: unweighted mean of the eight benchmark metrics on this slice, not on the full evaluation sets.

Endpoint	State step residual	Logit $D_{\cos}$ from H_2	Top-1 changed from H_2	Mean score
H_2	reference	0.0000	0.0000	0.651
H_3	0.1165	0.0457	0.1089	0.628
H_4	0.0228	0.0952	0.1536	0.616
H_{10}	0.0010	0.1882	0.2946	0.457
H_{20}	0.0004	0.1951	0.3670	0.447
H_{50}	0.0003	0.1933	0.3999	0.417

Takeaway: Extra H-module calls barely move the state yet hurt: by H_{10} the step size is 0.001, but 29% of top-1 predictions change and the slice mean falls from 0.651 to 0.457.

C.2 HRM-Text State Perturbations

Removing the Carry State. The main mechanism study varies the Stage 2 L-module depth while retaining the standard input state. Here we ask whether additional L-module calls can recover the trained path after its Stage 1 carry is removed, and whether the remaining difference affects the H_2 readout. We remove the z_L^3 state carried into Stage 2 while preserving the same z_H^1. The standard carry path c_j and the drop path d_j are

$$c_3 = z_L^3, \qquad d_3 = 0, \qquad c_j = L(c_{j-1} + z_H^1), \quad d_j = L(d_{j-1} + z_H^1), \quad j = 4, \ldots, 20.$$

At each L_j, we compare c_j directly with d_j using $d_{\cos}(c_j, d_j)$ and $d_{\text{rel}}(c_j, d_j)$, with the relative distance normalized by the carried state. After L_6 or L_{20}, we also compare the two H_2 readouts obtained by applying the same H-module to $z_H^1 + c_j$ and $z_H^1 + d_j$. Table 11 reports these distances over the same 200-example probe set and two Stage 1 initializations.

Table 11: Effect of removing the z_L^3 carry at the start of Stage 2. L_j rows compare c_j with d_j; H_2 rows compare readouts from $z_H^1 + c_j$ and $z_H^1 + d_j$. Top-1 changed is the fraction of differing readout predictions for the default Stage 1 initialization, over the 200 probe examples.

State	Cosine distance	Relative ℓ_2	Top-1 changed (default)
L_4	0.2008	0.6313	–
L_5	0.1030	0.4480	–
L_6	0.0832	0.3882	–
L_{20}	0.0372	0.2490	–
H_2 after L_6	0.0562	0.3061	21.0%
H_2 after L_{20}	0.0270	0.2102	10.5%

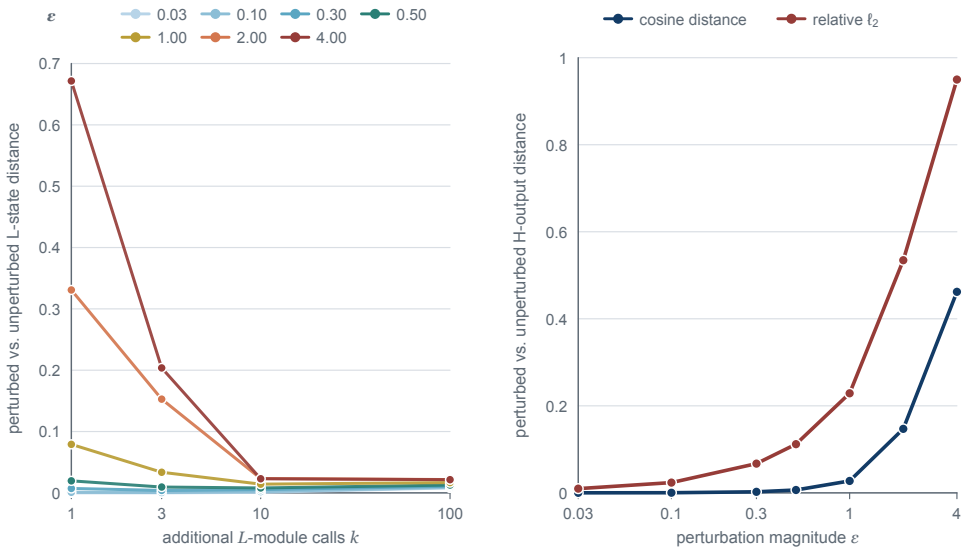
The two low-level paths draw closer with continued refinement: $d_{\cos}(c_4, d_4) = 0.2008$, falling to $d_{\cos}(c_{20}, d_{20}) = 0.0372$. They do not coincide at the trained depth, however: after $L_6 \to H_2$, the readouts have cosine distance 0.0562, 21.0% of diagnostic top-1 predictions change, and the benchmark score falls. Extending Stage 2 to L_{20} reduces the readout distance to 0.0270 and the top-1 change to 10.5%, while improving the drop path's teacher-forced score (scores not tabulated). Extra L-module calls thus partially repair the removed carry, but it still matters at the trained depth.

Takeaway: Continued L-module calls pull the carry-free path toward the default one (0.20 at L_4, 0.04 at L_{20}), yet at the trained depth the missing carry still changes 21% of top-1 predictions.

Recovery from Controlled Perturbations. The carry-state ablation creates one structured displacement. To test whether continued L-module refinement repairs a displaced state, we vary the perturbation magnitude at the trained Stage 1 endpoint:

$$z^3_{L,\text{pert}} = z^3_L + \delta, \qquad \|\delta\|_2 = \epsilon \|z^3_L\|_2. \tag{3}$$

From $z_L^3 + \delta$ and the unperturbed z_L^3, we run the same additional L-module calls under fixed z_H^0 and measure the cosine distance between the two resulting low-level states after each call. We separately compare the H_1 outputs obtained by applying one H-module call to the perturbed and unperturbed inputs. Figure 7 shows that the L-module paths realign: at $\epsilon = 4$, their cosine distance falls from 0.6716 after one call to 0.0230 after ten. The H-module output is also stable under smaller perturbations ($d_{\cos} = 0.0274$ at $\epsilon = 1$), but its distance reaches 0.4620 at $\epsilon = 4$.



(a) L-module trajectory recovery. **(b)** H_1-output stability.

Figure 7: Responses to a perturbation of z_L^3. (a) Cosine distance between perturbed and unperturbed low-level states after k additional L-module calls under the same z_H^0. (b) Cosine and relative ℓ_2 distances between the corresponding H_1 outputs after one H-module call.

Table 12: Distances between perturbed and unperturbed continuations from z_L^3. The H_1 columns compare one-call outputs; the L-module columns compare low-level states after k additional calls under fixed z_H^0.

	H_1 outputs		L-module states			
ϵ	$d_{\cos}$	d_{rel}	$k=1$	$k=3$	$k=10$	$k=100$
0.03	0.0000	0.0097	0.0001	0.0001	0.0008	0.0081
0.10	0.0003	0.0236	0.0009	0.0006	0.0020	0.0089
0.30	0.0024	0.0672	0.0072	0.0040	0.0053	0.0108
0.50	0.0066	0.1119	0.0196	0.0095	0.0080	0.0127
1.00	0.0274	0.2288	0.0791	0.0336	0.0142	0.0168
2.00	0.1471	0.5346	0.3308	0.1528	0.0234	0.0217
4.00	0.4620	0.9498	0.6716	0.2037	0.0230	0.0215

The carry removal and direct-perturbation experiments are complementary: continued L-module calls can repair a displaced state, although removing the carry changes the trained-depth output.

Takeaway: The L-module repairs even large displacements: at $\epsilon = 4$, perturbed and clean paths go from 0.67 to 0.02 within ten calls, while an immediate H-module call leaves them 0.46 apart.

Membership of the Default Endpoint. Recovery alone does not show whether alternative trajectories approach the same region as the default trajectory. We test this separately using 256 norm-matched alternative low-level initializations (128 directions shared across tokens and 128 token-wise random directions), not the z_L^3 perturbations above. For each input, let $h^{(0)}$ be the final H_2 state of the default trajectory and $h^{(1)}, \ldots, h^{(M)}$, with $M = 256$, the alternative endpoints. We compare the mean default-to-alternative endpoint distance with the mean pairwise distance among alternative endpoints:

$$D_{\text{def}} = \frac{1}{M}\sum_{i=1}^{M} d_{\cos}\left(h^{(0)}, h^{(i)}\right), \quad D_{\text{pert}} = \frac{2}{M(M-1)} \sum_{1 \le i < j \le M} d_{\cos}\left(h^{(i)}, h^{(j)}\right).$$

Table 13: Final H_2-state endpoint distances under alternative low-level initializations. D_{def} compares the default endpoint with the 256 alternatives; D_{pert} compares pairs of alternative endpoints. Cosine distances are averaged over 25 examples per benchmark (200 in total).

Benchmark	D_{def}	D_{pert}
ARC	0.00523	0.00484
BoolQ	0.01003	0.00996
DROP	0.00643	0.00561
GSM8K	0.00666	0.00651
HellaSwag	0.00802	0.00796
MATH	0.00424	0.00403
MMLU	0.01215	0.01145
Winogrande	0.01907	0.01644
Mean	0.00898	0.00835

Across all eight benchmarks, D_{def} is comparable to D_{pert}: their means are 0.00898 and 0.00835, respectively. Thus, the default trajectory's final H_2 state lies in the same narrow endpoint region as those from alternative initializations. This endpoint comparison complements, but does not directly measure, the L-state recovery experiment above, which perturbs z_L^3 rather than the initialization.

Takeaway: Final H_2 states from 256 alternative initializations lie nearly as close to the default endpoint (0.0090) as to each other (0.0084), so the default ends in the same narrow region.

C.3 Standard Looped Transformer: Refinement Without a Separate H Module

The standard looped Transformer isolates the recurrent part of the mechanism. It has one shared module F and reads logits directly from F_4, without a parameter-distinct terminal transformation. If basin refinement is a general property of recurrent depth, F should contract different starts and repair mid-trajectory perturbations on its own, without any separate terminal module.

Table 14: Standard looped TF contraction and continuation across 258 controlled initial states: the default and zero states plus 128 shared-random and 128 token-random directions. Cross-init spread is the mean pairwise cosine distance, step residual compares consecutive states, and drift is measured from the trained F_4 readout state; all are cosine distances.

State	Cross-init spread	Step residual	Drift from F_4
F_0	1.0001	–	–
F_4	0.0170	0.0577	0.0000
F_{10}	0.0070	0.0210	0.0234
F_{50}	0.0054	0.0141	0.0316

By the trained fourth call, unrelated initial directions occupy a narrow region. Later calls tighten the cross-initialization spread, but the shared state continues to drift: F_4 is a useful finite-depth readout rather than a strict fixed point, as its drift keeps growing from 0.0234 at F_{10} to 0.0316 at F_{50}.

Table 15: Recovery after replacing the trained F_4 state. Entries are cosine distances from the clean continuation at the same call, averaged over the 200-example probe set.

Intervention at F_4	Initial distance	At F_8	At F_{20}
Reset to zero	1.0000	0.0477	0.0155
Reset to default initialization	1.0411	0.0247	0.0007
Reset to shared-random state	1.0076	0.0239	0.0050
Reset to token-random state	1.0002	0.0274	0.0092

Several additional F calls recover from a complete state replacement. The eight-benchmark mean on the 200-example slice after reset-and-continue to F_8 remains between 0.573 and 0.581, compared

with 0.578 at the clean F_4 readout. The single-module model therefore exhibits the refinement and recovery role, but it has no separate H-like module: the LM head reads a useful intermediate point on the recurrent trajectory rather than the output of a separately parameterized terminal transformation.

Takeaway: A plain looped Transformer also refines and recovers: different starts converge to a spread of 0.017 by F_4, and a full state reset recovers within four calls with little change in the slice mean (0.573–0.581 vs. 0.578). Lacking a separate H-module, it reads out an intermediate state.

C.4 Mechanism Transfer to L^7H

The simplification in Section 4 assumes that the functional separation observed in HRM-Text survives after removing its two persistent latent streams. We test this directly on the final (epoch-4) L^7H checkpoint using the same 200-example slice as the main mechanism study. For each input, we follow the recurrent state from 258 controlled starts: the default and zero states, 128 shared-random directions, and 128 token-random directions. The random states are norm-matched to the default fixed initialization, and $y_k = H(s_k + x_0)$ denotes the terminal readout after k L-module calls.

Table 16: Mechanism diagnostics transferred from HRM-Text to L^7H, averaged over 200 examples. Cross-initialization spread measures contraction across controlled starts, step residual measures successive L-module updates, and the final rows compare the terminal H-module readout y_k with recurrent states s_k after k L-module calls from the default initialization.

Diagnostic	State or comparison	Cosine distance	Relative ℓ_2
Cross-initialization spread	s_0	1.0001	1.4143
	s_3	0.0879	0.4170
	s_7	0.0191	0.1877
	s_{50}	0.0149	0.1593
Step residual	$s_0 \to s_1$	0.9093	1.3263
	$s_2 \to s_3$	0.2538	0.7053
	$s_6 \to s_7$	0.0135	0.1581
	$s_{49} \to s_{50}$	0.0019	0.0422
Terminal transformation	s_7 vs. s_{50}	0.0237	0.2091
	y_7 vs. s_7	0.6788	1.1650
	y_7 vs. s_{50}	0.6854	1.1707
	y_6 vs. y_7	0.0070	0.1101

Repeated L-module calls sharply reduce cross-initialization spread and local motion; by s_7, the state is close to its long continuation at s_{50}. The terminal state y_7, however, remains far from both s_7 and s_{50}, while y_6 and y_7 are close. Thus the simplified architecture preserves the mechanism that motivated it: recurrent basin refinement followed by a stable, parameter-distinct final transformation.

Takeaway: L^7H reproduces both roles in one stream: the cross-initialization spread shrinks to 0.019 by s_7, while the H-module output stays about 0.68 away from the recurrent state.

Readout Dependence. We next test whether the H-module can replace refinement or tolerate an arbitrary input. Here y_7 is the trained readout state after seven L-module calls and one H-module call, which serves as the reference for every intervention below.

Table 17: Stress tests of the L^7H readout on the 200-example mechanism slice. Interventions change the number of L-module calls or remove the state passed to the H-module. Readout distance compares the resulting H-module state with the trained y_7 readout; Mean score: unweighted mean of the eight official benchmark metrics, computed on this slice rather than on the full evaluation sets.

Intervention	Readout cosine distance	Mean score
Read after L_5	0.0285	0.6141
Read after L_6	0.0070	0.6297
Trained L^7H	0.0000	0.6276
Read after L_{10}	0.0083	0.6188
Drop s_7 before the H-module	0.6167	0.2269
Reset s_7 before the H-module	0.7329	0.2229

The H-module is stable once the L-module has nearly reached the trained basin, but it cannot substitute for that basin: dropping or resetting s_7 causes a large state change and score collapse.

Takeaway: The H-module relies on the L-module state: reading after 6 or 10 calls barely moves the slice mean (0.630 and 0.619 vs. 0.628), but dropping or resetting s_7 cuts it to about 0.22.

Recovery Before Readout. We perturb s_7 with shared-random noise of relative magnitude ϵ and test whether extra L-module calls before the H-module move the readout back toward the unperturbed y_7 output, and how many extra calls this recovery takes as ϵ grows.

Table 18: Denoising a shared-random perturbation of magnitude ϵ to s_7 before the H-module. Columns either apply the H-module immediately or first perform the indicated number of additional L-module calls. Entries are cosine distances from the clean y_7 readout.

ϵ	Direct H-module call	1 extra L-module call	3 extra L-module calls	13 extra L-module calls
0.5	0.0483	0.0134	0.0120	0.0141
1.0	0.1995	0.0365	0.0327	0.0140
2.0	0.6315	0.4268	0.2160	0.0149
4.0	0.8754	0.6179	0.2961	0.0158

Small perturbations are tolerated directly by the H-module, whereas large displacements require additional refinement by the L-module. After 13 extra L-module calls, even $\epsilon = 4$ noise is mapped near the clean readout. Token-wise noise shows the same qualitative behavior (not tabulated). Together with the standard looped Transformer results, this identifies recovery as a property of the repeated refinement module, while the H-module only reads from the recovered region.

Takeaway: In this perturbation test, thirteen extra L-module calls bring even $\epsilon = 4$ noise to within 0.016 of the clean readout, while applying the H-module immediately leaves 0.88.

Combined Picture. Across the three architectures, the complementary interventions support one consistent account. Recurrent low-level modules contract broad state differences and repair large displacements over multiple calls. Removing the HRM carry state changes the trained-depth output, while continued refinement partially realigns the displaced path. A parameter-distinct high-level module then maps the refined region to a useful output state, but neither an arbitrary input nor indefinite high-level continuation reproduces the trained computation.

C.5 Input-Injection Ablation for L^7H

The L^7H recurrence reinjects x_0 at every L-module call and at the final H-module call. To isolate the contribution of these routes, we compare full reinjection with L-only and no-reinjection variants under the same truncated backward window and training recipe.

The no-reinjection forward recurrence uses the token representation only for initialization:

$$s_0 = x_0, \qquad s_k = L(s_{k-1}), \qquad y = H(s_n).$$

To compare it under the same short backward window, let stop-gradient satisfy $\mathrm{sg}(v) = v$ and $D_v\,\mathrm{sg}(v) = 0$. At each retained tail call we use the zero-valued straight-through bridge

$$\mathrm{STE}_{x_0}(u) = u + x_0 - \mathrm{sg}(x_0), \qquad \frac{\partial\,\mathrm{STE}_{x_0}(u)}{\partial x_0} = I.$$

This tail straight-through estimator (tail-STE) (Bengio et al., 2013) leaves the forward recurrence unchanged while supplying direct embedding-gradient routes within the retained tail; it is not the exact derivative of the no-reinjection recurrence but a bridge that routes gradients to the embeddings.

Table 19: Input-injection ablations for L^7H under matched training settings. "Full $L + H$" injects the input into both modules, "L only" removes the terminal injection, and "No reinjection" removes all reinjection of x_0 (into both modules) and uses the tail straight-through estimator (tail-STE). Mean is the unweighted average of the eight benchmark scores; all variants share data and training budget.

Variant	ARC	BoolQ	DROP	GSM8K	Hella.	MATH	MMLU	Wino.	Mean
Full $L + H$	**75.51**	**85.90**	**80.94**	**80.44**	**53.97**	**50.78**	**54.27**	65.35	**68.40**
L only	73.38	85.17	78.17	77.03	51.54	48.70	53.61	**66.30**	66.74
No reinjection (tail-STE)	70.65	83.70	75.94	76.19	44.62	47.82	53.94	64.80	64.71

Takeaway: Removing terminal injection lowers the mean by 1.66 points (68.40 to 66.74); also removing recurrent injection, which requires a tail straight-through bridge for embedding gradients, lowers it by a further 2.03 points (to 64.71), so this second step mixes two changes.

D Gradient Derivations and Additional Results

D.1 HRM-Text Gradient Derivation and Validation

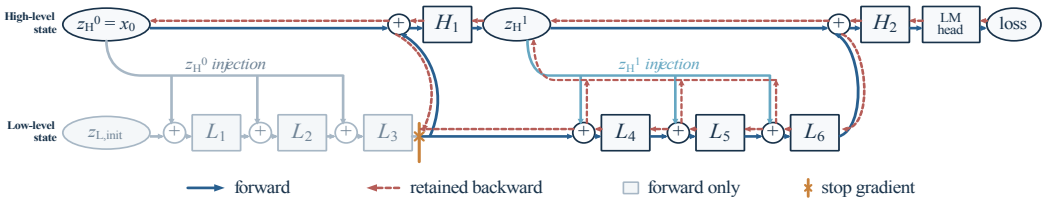


Figure 8: HRM-Text's $K = 5$ computation graph. Solid edges show the complete forward computation, dashed red edges show retained backward paths, and the marker after L_3 blocks gradients through the forward-only prefix. Calls of the same type share parameters.

D.1.1 Forward Graph and Notation

This appendix derives the gradients summarized in Section 5.1. We compare two computations for the same two-stage architecture: an idealized model in which each L-module stage is treated as an equilibrium, and the exact finite graph differentiated by HRM-Text after the backward-window warmup. The derivation first fixes a common forward graph, then traces how the two backward computations treat the same parameter paths through the H-module, the L-module, and the embeddings.

Both computations use the same shared modules and initial states:

$$\begin{aligned} f_L(\cdot\,;\theta_L), \quad f_H(\cdot\,;\theta_H), &\qquad \text{shared modules,} \\ z_H^0 = E_{\theta_E}(x), \quad z_L^0 = z_{L,\text{init}}, &\qquad \text{initial states.} \end{aligned}$$

Here f_L and f_H are the L- and H-module maps written L_{θ_L} and H_{θ_H} in Equation (1), and $E_{\theta_E}(x) = x_0$ is the token embedding. Under the trained $3L \to H \to 3L \to H$ schedule, the six L-module inputs are

$$\begin{aligned} u_1 &= z_L^0 + z_H^0, & u_2 &= z_L^1 + z_H^0, & u_3 &= z_L^2 + z_H^0, \\ u_4 &= z_L^3 + z_H^1, & u_5 &= z_L^4 + z_H^1, & u_6 &= z_L^5 + z_H^1, \end{aligned} \tag{4}$$

and the two H-module inputs are

$$a_1 = z_H^0 + z_L^3, \qquad a_2 = z_H^1 + z_L^6. \tag{5}$$

The complete numerical forward pass is therefore

$$\begin{aligned} z_L^1 &= f_L(z_L^0 + z_H^0; \theta_L), \\ z_L^2 &= f_L(z_L^1 + z_H^0; \theta_L), \\ z_L^3 &= f_L(z_L^2 + z_H^0; \theta_L), \\ z_H^1 &= f_H(z_H^0 + z_L^3; \theta_H), \\ z_L^4 &= f_L(z_L^3 + z_H^1; \theta_L), \\ z_L^5 &= f_L(z_L^4 + z_H^1; \theta_L), \\ z_L^6 &= f_L(z_L^5 + z_H^1; \theta_L), \\ z_H^2 &= f_H(z_H^1 + z_L^6; \theta_H), \end{aligned} \tag{6}$$

followed by $\mathcal{L} = \ell(z_H^2)$, where ℓ includes the language-model head and token loss.

To trace perturbations through this graph, the input Jacobians of L-module call i and H-module call t are

$$J_{L,i}^{\text{in}} = D_u f_L(u; \theta_L)|_{u=u_i}, \qquad J_{H,t}^{\text{in}} = D_a f_H(a; \theta_H)|_{a=a_t}, \tag{7}$$

while their direct parameter Jacobians are

$$J_{L,i}^{\theta} = D_{\theta_L} f_L(u; \theta_L)|_{u=u_i}, \qquad J_{H,t}^{\theta} = D_{\theta_H} f_H(a; \theta_H)|_{a=a_t}. \tag{8}$$

The first subscript names the module, the second indexes its call, and the superscript identifies the differentiation target. "Direct" means that the corresponding module input is held fixed. The token-embedding Jacobian is $J_E^{\theta} = D_{\theta_E} z_H^0$. Differentials are column vectors, so reverse-mode propagation applies the transposed Jacobians in reverse order. In particular, the order of every product below matters, since products of Jacobian matrices do not commute in general.

D.1.2 Implicit-Equilibrium Gradient

The equilibrium reference asks how changes in the conditioning state and shared parameters would propagate if each recurrent L-module stage were continued to a local equilibrium. We first derive the response of one stage and then compose the two stages used by HRM-Text.

One L-Module Stage. At stage t, the finite L-module calls are replaced by the equilibrium equation

$$z_{L,t}^{\star} = f_L(z_{L,t}^{\star} + z_H^{t-1}; \theta_L). \tag{9}$$

The summed input to the following H-module call is $a_t^{\star} = z_{L,t}^{\star} + z_H^{t-1}$. The corresponding local Jacobians are

$$\begin{aligned} J_{L,t}^{\text{in},\star} &= D_a f_L(a; \theta_L)\big|_{a=a_t^{\star}}, \\ J_{L,t}^{\theta,\star} &= D_{\theta_L} f_L(a_t^{\star}; \theta_L). \end{aligned} \tag{10}$$

Conditioning-State Response. To measure how the high-level conditioning state changes the equilibrium, we hold θ_L fixed and differentiate (9) with respect to z_H^{t-1}. Because $D_{z_H^{t-1}} a_t^{\star} = D_{z_H^{t-1}} z_{L,t}^{\star} + I$, the chain rule gives

$$\begin{aligned} D_{z_H^{t-1}} z_{L,t}^{\star} &= J_{L,t}^{\text{in},\star}\big(D_{z_H^{t-1}} z_{L,t}^{\star} + I\big), \\ \big(I - J_{L,t}^{\text{in},\star}\big) D_{z_H^{t-1}} z_{L,t}^{\star} &= J_{L,t}^{\text{in},\star}. \end{aligned} \tag{11}$$

When $I - J_{L,t}^{\text{in},\star}$ is invertible, the complete conditioning-state transport is

$$T_t^{\text{IFT}} = \big(I - J_{L,t}^{\text{in},\star}\big)^{-1}. \tag{12}$$

The equilibrium-state response is therefore

$$D_{z_H^{t-1}} z_{L,t}^\star = T_t^{\mathrm{IFT}} J_{L,t}^{\mathrm{in},\star},$$
$$D_{z_H^{t-1}} a_t^\star = I + D_{z_H^{t-1}} z_{L,t}^\star = I + T_t^{\mathrm{IFT}} J_{L,t}^{\mathrm{in},\star} = T_t^{\mathrm{IFT}}. \tag{13}$$

The final equality follows from $T_t^{\mathrm{IFT}}(I - J_{L,t}^{\mathrm{in},\star}) = I$.

Parameter Response. To isolate the contribution of the shared parameters of the L-module, we instead hold z_H^{t-1} fixed and differentiate (9) with respect to θ_L:

$$\begin{aligned} D_{\theta_L} z_{L,t}^\star &= J_{L,t}^{\theta,\star} + J_{L,t}^{\mathrm{in},\star} D_{\theta_L} z_{L,t}^\star, \\ \left(I - J_{L,t}^{\mathrm{in},\star}\right) D_{\theta_L} z_{L,t}^\star &= J_{L,t}^{\theta,\star}, \\ D_{\theta_L} z_{L,t}^\star &= T_t^{\mathrm{IFT}} J_{L,t}^{\theta,\star}, \\ D_{\theta_L} a_t^\star &= T_t^{\mathrm{IFT}} J_{L,t}^{\theta,\star} =: S_t^{\mathrm{IFT}}. \end{aligned} \tag{14}$$

Thus T_t^{IFT} always denotes transport from the conditioning state to the complete summed input passed to the stage-t H-module call, whereas S_t^{IFT} denotes its local sensitivity to θ_L with z_H^{t-1} held fixed. When z_H^{t-1} itself depends on θ_L, as in Stage 2, that cross-stage path is propagated separately below.

Two-Stage Forward Derivatives. We now compose the single-stage response across HRM-Text's two stages. In the idealized model,

$$z_H^t = f_H(z_H^{t-1} + z_{L,t}^\star; \theta_H), \qquad t \in \{1, 2\}.$$

The H-module Jacobians are evaluated at these equilibrium-conditioned inputs, and z_H^0 is independent of θ_H and θ_L because it depends only on the token embeddings.

Because θ_H enters only through the two H-module calls, its Stage 1 derivative is

$$D_{\theta_H} z_H^1 = J_{H,1}^\theta,$$

and the second stage gives

$$D_{\theta_H} z_H^2 = J_{H,2}^\theta + J_{H,2}^{\mathrm{in}} T_2^{\mathrm{IFT}} J_{H,1}^\theta. \tag{15}$$

The first term is the direct parameter derivative of H_2; the second propagates the direct H_1 derivative through the complete Stage 2 equilibrium response T_2^{IFT}, including its direct identity path.

The shared parameters of the L-module affect the Stage 1 equilibrium through

$$D_{\theta_L} z_H^1 = J_{H,1}^{\mathrm{in}} S_1^{\mathrm{IFT}}.$$

At Stage 2, this earlier contribution joins the direct effect on the local L-module equilibrium:

$$\begin{aligned} D_{\theta_L} z_H^2 &= J_{H,2}^{\mathrm{in}} \left(T_2^{\mathrm{IFT}} D_{\theta_L} z_H^1 + S_2^{\mathrm{IFT}}\right) \\ &= J_{H,2}^{\mathrm{in}} \left(T_2^{\mathrm{IFT}} J_{H,1}^{\mathrm{in}} S_1^{\mathrm{IFT}} + S_2^{\mathrm{IFT}}\right). \end{aligned} \tag{16}$$

This exposes one shared-parameter contribution from each L-module stage.

The embedding perturbation follows the same two equilibrium responses. It first reaches Stage 1 through

$$D_{\theta_E} z_H^1 = J_{H,1}^{\mathrm{in}} T_1^{\mathrm{IFT}} J_E^\theta,$$

and then propagates through Stage 2:

$$D_{\theta_E} z_H^2 = J_{H,2}^{\mathrm{in}} T_2^{\mathrm{IFT}} J_{H,1}^{\mathrm{in}} T_1^{\mathrm{IFT}} J_E^\theta. \tag{17}$$

The embedding path passes through both equilibrium responses.

Reverse-Mode Loss Gradient. The forward derivatives identify the paths, while the experimental reference is computed in reverse mode. Starting from $q_{H,2} = \nabla_{z_H^2}\mathcal{L}$, we avoid forming either inverse T_t^{IFT} and instead solve two adjoint systems:

$$\begin{aligned} q_{a,2} &= \left(J_{H,2}^{\mathrm{in}}\right)^\top q_{H,2}, & \left(I - (J_{L,2}^{\mathrm{in},\star})^\top\right) q_{H,1}^{\mathrm{IFT}} &= q_{a,2}, \\ q_{a,1}^{\mathrm{IFT}} &= \left(J_{H,1}^{\mathrm{in}}\right)^\top q_{H,1}^{\mathrm{IFT}}, & \left(I - (J_{L,1}^{\mathrm{in},\star})^\top\right) q_{H,0}^{\mathrm{IFT}} &= q_{a,1}^{\mathrm{IFT}}. \end{aligned} \tag{18}$$

Equivalently,

$$q_{H,1}^{\mathrm{IFT}} = \left(T_2^{\mathrm{IFT}}\right)^\top q_{a,2}, \qquad q_{H,0}^{\mathrm{IFT}} = \left(T_1^{\mathrm{IFT}}\right)^\top q_{a,1}^{\mathrm{IFT}}.$$

Transposing Equations (15)–(17) gives

$$\begin{aligned} \nabla_{\theta_H}^{\mathrm{IFT}}\mathcal{L} &= \left(J_{H,2}^{\theta}\right)^\top q_{H,2} + \left(J_{H,1}^{\theta}\right)^\top q_{H,1}^{\mathrm{IFT}}, \\ \nabla_{\theta_L}^{\mathrm{IFT}}\mathcal{L} &= \left(S_2^{\mathrm{IFT}}\right)^\top q_{a,2} + \left(S_1^{\mathrm{IFT}}\right)^\top q_{a,1}^{\mathrm{IFT}}, \\ \nabla_{\theta_E}^{\mathrm{IFT}}\mathcal{L} &= \left(J_E^{\theta}\right)^\top q_{H,0}^{\mathrm{IFT}}. \end{aligned} \tag{19}$$

These are the IFT-style parameter gradients used as the local reference in the main-text comparison.

D.1.3 Exact Gradient of the $K = 5$ Stopped Graph

The comparison requires an equally explicit expression for the gradient that HRM-Text actually uses. We therefore keep the complete eight-call forward trajectory and trace only the five module calls retained by its stopped backward graph after warmup (Figure 8).

Differentiable Graph. The numerical forward pass remains unchanged from (6). After warmup, however, the value produced by the first three L-module calls is detached:

$$\bar{z}_L^3 = \mathrm{stopgrad}(z_L^3), \qquad D_\theta \bar{z}_L^3 = 0. \tag{20}$$

The differentiable graph is therefore

$$\begin{aligned} z_H^1 &= f_H(z_H^0 + \bar{z}_L^3; \theta_H), \\ z_L^4 &= f_L(\bar{z}_L^3 + z_H^1; \theta_L), \\ z_L^5 &= f_L(z_L^4 + z_H^1; \theta_L), \\ z_L^6 &= f_L(z_L^5 + z_H^1; \theta_L), \\ z_H^2 &= f_H(z_H^1 + z_L^6; \theta_H). \end{aligned} \tag{21}$$

Thus $K = 5$ describes the number of retained module calls $(H_1, L_4, L_5, L_6, H_2)$, not a different numerical forward trajectory; the forward values are identical.

Finite Chain Rule. To identify which paths remain, $D_\theta f_M^j$ ($M \in \{L, H\}$) denotes the direct derivative of call j with respect to a parameter block θ, holding the call's input fixed; it is zero when call j does not use θ. Differentiating Equation (21) gives the exact recurrences

$$\begin{aligned} D_\theta z_H^1 &= D_\theta f_H^1 + J_{H,1}^{\mathrm{in}} D_\theta z_H^0, \\ D_\theta z_L^4 &= D_\theta f_L^4 + J_{L,4}^{\mathrm{in}} D_\theta z_H^1, \\ D_\theta z_L^5 &= D_\theta f_L^5 + J_{L,5}^{\mathrm{in}}(D_\theta z_L^4 + D_\theta z_H^1), \\ D_\theta z_L^6 &= D_\theta f_L^6 + J_{L,6}^{\mathrm{in}}(D_\theta z_L^5 + D_\theta z_H^1), \\ D_\theta z_H^2 &= D_\theta f_H^2 + J_{H,2}^{\mathrm{in}}(D_\theta z_H^1 + D_\theta z_L^6). \end{aligned} \tag{22}$$

To isolate propagation from the incoming state z_H^1, temporarily hold every parameter fixed and perturb z_H^1 by dh. Successive application of the recurrences gives

$$\begin{aligned} dz_L^4 &= J_{L,4}^{\mathrm{in}}\, dh, \\ dz_L^5 &= J_{L,5}^{\mathrm{in}}(I + J_{L,4}^{\mathrm{in}})\, dh, \\ dz_L^6 &= J_{L,6}^{\mathrm{in}}(I + J_{L,5}^{\mathrm{in}} + J_{L,5}^{\mathrm{in}} J_{L,4}^{\mathrm{in}})\, dh. \end{aligned} \tag{23}$$

Because $a_2 = z_H^1 + z_L^6$, these paths combine into the finite Stage 2 response

$$\begin{aligned} da_2 &= T_2^{(5)}\, dh, \\ T_2^{(5)} &= I + J_{L,6}^{\text{in}} + J_{L,6}^{\text{in}} J_{L,5}^{\text{in}} + J_{L,6}^{\text{in}} J_{L,5}^{\text{in}} J_{L,4}^{\text{in}}. \end{aligned} \tag{24}$$

Thus $T_2^{(5)}$ maps a perturbation of z_H^1 to the complete H_2 input a_2 through the three retained L-module calls and through the direct identity route from z_H^1 into a_2.

The direct Stage 2 L-module parameter injections follow a different chain:

$$\begin{aligned} D_{\theta_L} z_L^4 &= J_{L,4}^{\theta}, \\ D_{\theta_L} z_L^5 &= J_{L,5}^{\theta} + J_{L,5}^{\text{in}} J_{L,4}^{\theta}, \\ D_{\theta_L} z_L^6 &= J_{L,6}^{\theta} + J_{L,6}^{\text{in}} J_{L,5}^{\theta} + J_{L,6}^{\text{in}} J_{L,5}^{\text{in}} J_{L,4}^{\theta}. \end{aligned} \tag{25}$$

Collecting these three parameter-injection paths gives

$$S_2^{(5)} = J_{L,6}^{\theta} + J_{L,6}^{\text{in}} J_{L,5}^{\theta} + J_{L,6}^{\text{in}} J_{L,5}^{\text{in}} J_{L,4}^{\theta}. \tag{26}$$

The difference between $T_2^{(5)}$ and $S_2^{(5)}$ is structural. $T_2^{(5)}$ maps an incoming hidden-state perturbation and includes the direct identity route to H_2; $S_2^{(5)}$ maps a shared L-module parameter perturbation injected inside the three tracked L-module calls and has no identity term.

Parameter and Loss Gradients. The state transport $T_2^{(5)}$ and parameter sensitivity $S_2^{(5)}$ now expose the retained path for each parameter group. Substituting them into Equation (22) gives the shared H-module response

$$D_{\theta_H} z_H^2 = J_{H,2}^{\theta} + J_{H,2}^{\text{in}} T_2^{(5)} J_{H,1}^{\theta}. \tag{27}$$

The shared L-module parameter response is

$$D_{\theta_L} z_H^2 = J_{H,2}^{\text{in}} S_2^{(5)} = J_{H,2}^{\text{in}} \big(J_{L,6}^{\theta} + J_{L,6}^{\text{in}} J_{L,5}^{\theta} + J_{L,6}^{\text{in}} J_{L,5}^{\text{in}} J_{L,4}^{\theta} \big). \tag{28}$$

The embedding response is

$$D_{\theta_E} z_H^2 = J_{H,2}^{\text{in}} T_2^{(5)} J_{H,1}^{\text{in}} J_E^{\theta}. \tag{29}$$

Here $D_{\theta_L} z_H^1 = 0$ because $D_{\theta_L} \bar{z}_L^3 = 0$. The first three L-module calls therefore transmit neither the shared-L-module derivative nor the embedding derivative; the embeddings still reach the loss through the direct $z_H^0 \to H_1$ path.

For compact reverse-mode expressions, write

$$q_{H,2} = \nabla_{z_H^2} \mathcal{L}, \qquad q_{a,2} = \big(J_{H,2}^{\text{in}}\big)^\top q_{H,2}, \qquad q_{H,1}^{(5)} = \big(T_2^{(5)}\big)^\top q_{a,2}, \qquad q_{H,0}^{(5)} = \big(J_{H,1}^{\text{in}}\big)^\top q_{H,1}^{(5)}.$$

Transposing Equations (27)–(29) gives the exact loss gradients of the stopped graph:

$$\begin{aligned} \nabla_{\theta_H}^{(5)} \mathcal{L} &= \big(J_{H,2}^{\theta}\big)^\top q_{H,2} + \big(J_{H,1}^{\theta}\big)^\top q_{H,1}^{(5)}, \\ \nabla_{\theta_L}^{(5)} \mathcal{L} &= \big(S_2^{(5)}\big)^\top q_{a,2} \\ &= \big(J_{L,6}^{\theta}\big)^\top q_{a,2} + \big(J_{L,5}^{\theta}\big)^\top \big(J_{L,6}^{\text{in}}\big)^\top q_{a,2} \\ &\quad + \big(J_{L,4}^{\theta}\big)^\top \big(J_{L,5}^{\text{in}}\big)^\top \big(J_{L,6}^{\text{in}}\big)^\top q_{a,2}, \\ \nabla_{\theta_E}^{(5)} \mathcal{L} &= \big(J_E^{\theta}\big)^\top q_{H,0}^{(5)}. \end{aligned} \tag{30}$$

These are exact gradients of the implemented stopped graph, not gradients of the complete eight-call graph. The θ_H gradient is an exception: it coincides with the complete-graph gradient, because θ_H enters only at H_1 and H_2, downstream of the stop, with all loss paths retained. Hence, at every state, the H-module entries at $K = 5$ equal $1.000/1.000$ in the complete-graph comparison of Table 22.

D.1.4 Path-by-Path Comparison

With both gradients written on the same forward graph, we can now compare what truncation preserves and removes for each parameter group: the H-module, the L-module, and the embeddings.

H-Module Parameters. The H-module parameters receive a direct contribution at H_2 and a contribution propagated from H_1. The two gradients are

$$\nabla^{\mathrm{IFT}}_{\theta_H}\mathcal{L} = \big(J^{\theta}_{H,2}\big)^{\top} q_{H,2} + \big(J^{\theta}_{H,1}\big)^{\top} q^{\mathrm{IFT}}_{H,1},$$
$$\nabla^{(5)}_{\theta_H}\mathcal{L} = \big(J^{\theta}_{H,2}\big)^{\top} q_{H,2} + \big(J^{\theta}_{H,1}\big)^{\top} q^{(5)}_{H,1}. \tag{31}$$

The direct H_2 contribution is identical. The only structural change to the propagated H_1 contribution is

$$q^{\mathrm{IFT}}_{H,1} \quad \longrightarrow \quad q^{(5)}_{H,1},$$
$$q^{\mathrm{IFT}}_{H,1} = \big(T^{\mathrm{IFT}}_2\big)^{\top} q_{a,2} = \big[\big(I - J^{\mathrm{in},\star}_{L,2}\big)^{-1}\big]^{\top} q_{a,2},$$
$$q^{(5)}_{H,1} = \big(T^{(5)}_2\big)^{\top} q_{a,2} = \big[I + J^{\mathrm{in}}_{L,6} + J^{\mathrm{in}}_{L,6}J^{\mathrm{in}}_{L,5} + J^{\mathrm{in}}_{L,6}J^{\mathrm{in}}_{L,5}J^{\mathrm{in}}_{L,4}\big]^{\top} q_{a,2}.$$

No Stage 1 equilibrium factor is needed: all three Stage 1 L-module calls precede the first use of θ_H, so perturbing θ_H cannot affect them in either of the two computations.

L-Module Parameters. The shared parameters of the L-module differ more substantially because the equilibrium reference contains one contribution from each stage:

$$\nabla^{\mathrm{IFT}}_{\theta_L}\mathcal{L} = \big(S^{\mathrm{IFT}}_2\big)^{\top} q_{a,2} + \big(S^{\mathrm{IFT}}_1\big)^{\top} q^{\mathrm{IFT}}_{a,1}. \tag{32}$$

The finite gradient retains only the three Stage 2 parameter injections:

$$\nabla^{(5)}_{\theta_L}\mathcal{L} = \big(S^{(5)}_2\big)^{\top} q_{a,2}. \tag{33}$$

The first term in the IFT-style expression is the Stage 2 equilibrium contribution, while the second carries the Stage 1 contribution through both stages. Truncation replaces the former by

$$S^{\mathrm{IFT}}_2 \quad \longrightarrow \quad S^{(5)}_2,$$
$$S^{\mathrm{IFT}}_2 = T^{\mathrm{IFT}}_2 J^{\theta,\star}_{L,2} = \big(I - J^{\mathrm{in},\star}_{L,2}\big)^{-1} J^{\theta,\star}_{L,2},$$
$$S^{(5)}_2 = J^{\theta}_{L,6} + J^{\mathrm{in}}_{L,6}J^{\theta}_{L,5} + J^{\mathrm{in}}_{L,6}J^{\mathrm{in}}_{L,5}J^{\theta}_{L,4},$$

not simply $T^{\mathrm{IFT}}_2 \to T^{(5)}_2$. State propagation and direct parameter injection are different derivatives. The complete Stage 1 contribution is removed because $D_{\theta_L}\bar{z}^3_L = 0$, which implies $D_{\theta_L} z^1_H = 0$.

Embedding Parameters. To see how truncation changes embedding learning, we trace the two paths from z^0_H to the loss. The equilibrium reference propagates through both recurrent responses, whereas the finite graph keeps a direct Stage 1 route and the three-call Stage 2 response:

$$\nabla^{\mathrm{IFT}}_{\theta_E}\mathcal{L} = \big(J^{\theta}_{E}\big)^{\top} q^{\mathrm{IFT}}_{H,0},$$
$$\nabla^{(5)}_{\theta_E}\mathcal{L} = \big(J^{\theta}_{E}\big)^{\top} q^{(5)}_{H,0}. \tag{34}$$

The structural change to the propagated embedding cotangent is

$$q^{\mathrm{IFT}}_{H,0} \quad \longrightarrow \quad q^{(5)}_{H,0},$$
$$q^{\mathrm{IFT}}_{H,0} = \big(T^{\mathrm{IFT}}_1\big)^{\top}\big(J^{\mathrm{in}}_{H,1}\big)^{\top}\big(T^{\mathrm{IFT}}_2\big)^{\top} q_{a,2},$$
$$q^{(5)}_{H,0} = \big(J^{\mathrm{in}}_{H,1}\big)^{\top}\big(T^{(5)}_2\big)^{\top} q_{a,2},$$
$$T^{\mathrm{IFT}}_t = \big(I - J^{\mathrm{in},\star}_{L,t}\big)^{-1}, \qquad t \in \{1,2\},$$
$$T^{(5)}_2 = I + J^{\mathrm{in}}_{L,6} + J^{\mathrm{in}}_{L,6}J^{\mathrm{in}}_{L,5} + J^{\mathrm{in}}_{L,6}J^{\mathrm{in}}_{L,5}J^{\mathrm{in}}_{L,4}.$$

Thus the IFT-style gradient contains the Stage 1 and Stage 2 equilibrium transports. The finite gradient keeps the three-call Stage 2 transport, while Stage 1 reduces to the direct $z^0_H \to H_1$ route. The embedding gradient is therefore retained, but its feedback routes through L_1, L_2, L_3 are removed.

D.1.5 Relation to the Neumann Series

The path comparison shows which terms truncation removes; the Neumann view explains why the retained terms can nevertheless point in a similar direction. Following analyses of recurrent backpropagation (Pineda, 1987; Liao et al., 2018) and implicit layers (Bai et al., 2019), if the Stage 2 equilibrium is locally stable, $\rho(J^{\mathrm{in},\star}_{L,2}) < 1$, its response expands as

$$T_2^{\mathrm{IFT}} = \left(I - J^{\mathrm{in},\star}_{L,2}\right)^{-1} = I + J^{\mathrm{in},\star}_{L,2} + \left(J^{\mathrm{in},\star}_{L,2}\right)^2 + \left(J^{\mathrm{in},\star}_{L,2}\right)^3 + \cdots . \tag{35}$$

If the finite Stage 2 input Jacobians are locally similar, $J^{\mathrm{in}}_{L,4} \approx J^{\mathrm{in}}_{L,5} \approx J^{\mathrm{in}}_{L,6} \approx \bar{J}^{\mathrm{in}}_L$, then

$$T_2^{(5)} \approx I + \bar{J}^{\mathrm{in}}_L + (\bar{J}^{\mathrm{in}}_L)^2 + (\bar{J}^{\mathrm{in}}_L)^3. \tag{36}$$

The propagated hidden-state response therefore resembles a four-term Neumann truncation.

The parameter path has one fewer term. If the finite and reference parameter Jacobians are locally represented by the same $\bar{J}^{\theta}_L$, then

$$\begin{aligned} S_2^{(5)} &\approx \bar{J}^{\theta}_L + \bar{J}^{\mathrm{in}}_L \bar{J}^{\theta}_L + (\bar{J}^{\mathrm{in}}_L)^2 \bar{J}^{\theta}_L, \\ S_2^{\mathrm{IFT}} &\approx \bar{J}^{\theta}_L + \bar{J}^{\mathrm{in}}_L \bar{J}^{\theta}_L + (\bar{J}^{\mathrm{in}}_L)^2 \bar{J}^{\theta}_L + (\bar{J}^{\mathrm{in}}_L)^3 \bar{J}^{\theta}_L + \cdots . \end{aligned} \tag{37}$$

The difference occurs because the preceding high-level state enters the H_2 input directly, producing the identity term in $T_2^{(5)}$, whereas θ_L enters only through the three tracked L-module calls.

This analogy does not turn the implemented gradient into an IFT gradient. The finite Jacobians are evaluated at different states, the forward pass does not solve an exact fixed point, Stage 2 begins from the carried z^3_L, contraction is not guaranteed, and the Stage 1 L-module parameter path is stopped. The Neumann view therefore supplies a local comparison, not an equality.

D.1.6 Probe and Numerical Diagnostics

The preceding derivation supplies a local theoretical reference. We next check whether the states and numerical solves used to construct that reference are sufficiently stable for directional comparison. The fixed-point probe uses ten matched batches from the 0.69B HRM-Text checkpoint at initialization and after epochs 1–3; the adjoint-solver diagnostics cover all three architectures at the same four model states. The IFT-style reference is linearized at the actual z^3_L and z^6_L states and computed through matrix-free adjoint solves. We also verify the analytic expressions in Equation (30) against direct automatic differentiation of the stopped graph; the maximum relative discrepancy is 0.0091.

Given a candidate state z conditioned on h, one additional L-module call produces $\widehat{z} = L(z + h)$. Its directional, relative, and norm changes are

$$d_{\cos}(z, \widehat{z}) = 1 - \frac{\langle \widehat{z}, z \rangle}{\|\widehat{z}\|_2 \|z\|_2}, \qquad d_{\mathrm{rel}}(z, \widehat{z}) = \frac{\|\widehat{z} - z\|_2}{\|z\|_2}, \qquad s_{\mathrm{norm}}(z) = \frac{\|\widehat{z}\|_2}{\|z\|_2}. \tag{38}$$

An exact one-step fixed point has $d_{\cos} = d_{\mathrm{rel}} = 0$ and $s_{\mathrm{norm}} = 1$. Table 20 reports the measurements at every model state used by the gradient probe (initialization and epochs 1–3).

Table 20: One-step fixed-point diagnostics at the actual three-call states. Cosine and relative-distance entries are mean ± population standard deviation over ten matched batches. Norm ratios round to 1.0000 because each complete L-module stack ends with RMSNorm.

Model state	Candidate state	$d_{\cos}$	d_{rel}	s_{norm}
Scratch	Stage 1 z^3_L	0.0725 ± 0.0001	0.3809 ± 0.0002	1.0000
Scratch	Stage 2 z^6_L	0.0274 ± 0.0001	0.2340 ± 0.0003	1.0000
Epoch 1	Stage 1 z^3_L	0.0988 ± 0.0018	0.4445 ± 0.0040	1.0000
Epoch 1	Stage 2 z^6_L	0.0100 ± 0.0003	0.1411 ± 0.0024	1.0000
Epoch 2	Stage 1 z^3_L	0.0852 ± 0.0016	0.4128 ± 0.0038	1.0000
Epoch 2	Stage 2 z^6_L	0.0739 ± 0.0014	0.3843 ± 0.0037	1.0000
Epoch 3	Stage 1 z^3_L	0.1059 ± 0.0017	0.4602 ± 0.0038	1.0000
Epoch 3	Stage 2 z^6_L	0.0961 ± 0.0020	0.4384 ± 0.0046	1.0000

The states are therefore not strict one-step fixed points. The closest case is Stage 2 at epoch 1, with cosine distance 0.0100 and relative distance 0.1411. The nearly identical norm ratios are a consequence of the final RMSNorm and are only a scale check, not evidence of convergence.

Adjoint Solver Quality. We solve each adjoint system with restarted GMRES (Saad & Schultz, 1986), applying $A(v) = (I - J^\top)v$ through matrix-free vector-Jacobian products. Solver accuracy is measured by the true relative residual

$$\varepsilon_{\text{res}}(\widehat{v}) = \frac{\|A\widehat{v} - b\|_2}{\|b\|_2}, \tag{39}$$

while within-architecture budget sensitivity is measured by the cosine similarity and relative ℓ_2 difference between the 32-matvec solution and a higher-budget solution. Table 21 summarizes these diagnostics over the four model states used in the gradient analysis. Solves use a 32-matvec budget, restart size 8, and target relative residual 10^{-3}. Residual ranges span checkpoint-level means; the higher-budget comparison uses one matched batch per model state and, for HRM-Text, per stage.

Table 21: Adjoint-solver diagnostics under a common 32-matvec budget. The residual measures linear-system error; lower is better. The last two columns compare the gradient direction with the higher-budget solution, where cosine near one and relative ℓ_2 near zero indicate numerical stability.

Architecture	32-matvec residual	Higher budget	Direction cosine	Relative ℓ_2
HRM-Text	0.126–0.512	48	0.9968–0.9998	0.023–0.081
L^7H	0.125–0.416	64	0.9943–0.9991	0.042–0.109
Standard Looped TF	0.115–0.657	64	0.9897–0.9997	0.026–0.146

The matched-budget residuals remain above this 10^{-3} target. Within each architecture, however, the higher-budget comparison shows that the computed direction changes little with additional solver work. These checks support using the local implicit responses as stable directional references for an equilibrium-motivated comparison, but they neither compare solver quality across architectures nor imply that the finite three-call forward has reached an exact fixed point.

Takeaway: The probed states are not exact fixed points (one-step cosine distance 0.01–0.11), and adjoint residuals stay above the 10^{-3} target (0.12–0.66), but the computed IFT-style directions change little with more solver work (direction cosine ≥ 0.989). The reference is therefore a stable directional comparison, not an exact equilibrium gradient of the finite model.

D.2 Gradient Agreement Across Architectures

The same protocol underlies all three architecture comparisons: the model state is sampled at initialization and after epochs 1–3, each entry averages the same ten matched batches, and every cell reports cosine c / norm ratio r. The numerical backward-window ranges remain architecture-specific because the executed recurrent graphs have different lengths.

D.2.1 HRM-Text Across Backward Windows

The main text reports only the $K = 5$ results after epochs 1–3 (Table 3). Table 22 gives the complete $K = 2, \ldots, 5$ grid under both references at all four model states, including initialization. Following HRM-Text's implementation, each window allocates gradients to H-module calls first: $K = 2$: $\{L_6, H_2\}$; $K = 3$: $\{H_1, L_6, H_2\}$; $K = 4$: $\{H_1, L_5, L_6, H_2\}$; $K = 5$: $\{H_1, L_4, L_5, L_6, H_2\}$.

Table 22: Complete HRM-Text backward-window sweep. For each model state and $K = 2, \ldots, 5$, the truncated gradient is compared with both exact full eight-call backpropagation and the IFT-style local reference. Each cell reports cosine similarity c and norm ratio r.

		Full eight-call BP			IFT-style local reference		
State	K	H-module	L-module	Embed.	H-module	L-module	Embed.
Scratch	2	0.669/0.468	0.315/0.143	–	0.646/0.460	0.400/0.197	–
	3	0.965/0.882	0.315/0.143	0.566/0.330	0.916/0.867	0.400/0.197	0.644/0.528
	4	0.992/0.952	0.367/0.217	0.600/0.374	0.943/0.936	0.457/0.298	0.682/0.598
	5	1.000/1.000	0.413/0.266	0.620/0.403	0.950/0.983	0.494/0.365	0.699/0.644
Epoch 1	2	0.932/0.857	0.723/0.531	–	0.935/0.860	0.748/0.561	–
	3	0.991/0.978	0.723/0.531	0.592/0.457	0.991/0.981	0.748/0.561	0.788/0.664
	4	0.998/0.992	0.835/0.724	0.634/0.510	0.995/0.996	0.831/0.765	0.816/0.740
	5	1.000/1.000	0.877/0.813	0.653/0.542	0.993/1.004	0.832/0.860	0.815/0.787
Epoch 2	2	0.918/0.851	0.687/0.516	–	0.932/0.867	0.665/0.624	–
	3	0.984/0.968	0.687/0.516	0.504/0.394	0.987/0.986	0.665/0.624	0.825/0.817
	4	0.996/0.988	0.813/0.720	0.558/0.457	0.987/1.006	0.749/0.870	0.832/0.947
	5	1.000/1.000	0.860/0.810	0.581/0.498	0.981/1.019	0.750/0.979	0.810/1.033
Epoch 3	2	0.918/0.857	0.654/0.494	–	0.939/0.881	0.634/0.679	–
	3	0.983/0.968	0.654/0.494	0.443/0.344	0.989/0.995	0.634/0.679	0.835/0.846
	4	0.996/0.988	0.777/0.693	0.491/0.404	0.986/1.015	0.708/0.953	0.829/0.993
	5	1.000/1.000	0.824/0.783	0.513/0.442	0.980/1.028	0.703/1.077	0.804/1.087

Table 23 calibrates the IFT-style reference against exact full finite backpropagation and shows the corresponding native truncated comparison at each model state. The full gradient remains positively aligned with the local reference, but the $K = 5$ gradient is more closely aligned for the L-module and embeddings after training (e.g., embeddings 0.815 vs. 0.645 at epoch 1).

Table 23: Native truncated ($K = 5$) and exact full finite ($K = 8$) gradients against the same IFT-style local reference. Each cell reports cosine c / norm ratio r, averaged over ten matched batches.

State	K	H-module c/r	L-module c/r	Embed. c/r
Scratch	5	0.950/0.983	0.494/0.365	0.699/0.644
	8 (full)	0.950/0.983	0.502/1.375	0.706/1.599
Epoch 1	5	0.993/1.004	0.832/0.860	0.815/0.787
	8 (full)	0.993/1.004	0.766/1.059	0.645/1.460
Epoch 2	5	0.981/1.019	0.750/0.979	0.810/1.033
	8 (full)	0.981/1.019	0.681/1.209	0.556/2.078
Epoch 3	5	0.980/1.028	0.703/1.077	0.804/1.087
	8 (full)	0.980/1.028	0.605/1.376	0.471/2.469

Table 24 isolates the central operator comparison at $K = 5$. Training initially raises the state-cotangent cosine from 0.636 at initialization to 0.932 after epoch 1. It then declines to 0.886 and 0.867 after epochs 2 and 3, while its projection onto the IFT-style direction stays close to one. The H-module gradient agrees more strongly because both gradients contain the identical direct H_2 term.

Table 24: Path-level $K = 5$ comparison with the IFT-style reference. "State" compares $(T_2^{(5)})^\top q_{a,2}$ with $(T_2^{\mathrm{IFT}})^\top q_{a,2}$; the rest compare parameter gradients. "State norm" and "State proj." are $\|a\|_2/\|b\|_2$ and $\langle a, b\rangle/\|b\|_2^2$ for the $K = 5$ cotangent a and IFT-style reference b. Ten batches per state.

State	State cosine	State norm	State proj.	H-module cosine	L-module cosine	Embed. cosine
Scratch	0.636	0.639	0.406	0.950	0.494	0.699
Epoch 1	0.932	1.022	0.951	0.993	0.832	0.815
Epoch 2	0.886	1.092	0.967	0.981	0.750	0.810
Epoch 3	0.867	1.158	1.004	0.980	0.703	0.804

D.2.2 GRADIENT SWEEPS FOR L^7H AND THE STANDARD LOOPED TRANSFORMER

Tables 25 and 27 report the complete finite-backpropagation sweeps summarized in Table 3. Tables 26 and 28 use the same states, windows, parameter groups, and metrics against each architecture's IFT-style local reference; Table 3 in the main text summarizes the native windows.

Table 25: Complete L^7H gradient sweep against full $K = 8$ finite backpropagation for the recurrent module and embeddings. Each cell reports cosine similarity c and norm ratio r between the K-window and full gradients, averaged over ten matched batches.

State	Group	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 6$	$K = 7$	$K = 8$
Scratch	L-module	0.547/0.311	0.678/0.487	0.772/0.619	0.842/0.726	0.900/0.817	0.949/0.899	1.000/1.000
	Embeddings	0.675/0.521	0.778/0.628	0.854/0.717	0.910/0.794	0.952/0.864	0.982/0.929	1.000/1.000
Epoch 1	L-module	0.765/0.560	0.882/0.748	0.930/0.854	0.952/0.921	0.966/0.958	0.981/0.980	1.000/1.000
	Embeddings	0.855/0.768	0.914/0.844	0.944/0.896	0.960/0.934	0.971/0.959	0.992/0.987	1.000/1.000
Epoch 2	L-module	0.661/0.482	0.780/0.664	0.833/0.765	0.863/0.830	0.915/0.901	0.975/0.973	1.000/1.000
	Embeddings	0.757/0.675	0.823/0.751	0.861/0.807	0.890/0.850	0.948/0.916	0.989/0.978	1.000/1.000
Epoch 3	L-module	0.495/0.358	0.596/0.506	0.667/0.657	0.699/0.683	0.748/0.738	0.835/0.817	1.000/1.000
	Embeddings	0.701/0.625	0.767/0.700	0.809/0.771	0.848/0.820	0.891/0.873	0.932/0.916	1.000/1.000

Table 26: Complete L^7H gradient sweep against the IFT-style local reference. Each cell reports cosine c / norm ratio r, averaged over ten matched batches.

State	Group	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 6$	$K = 7$	$K = 8$
Scratch	L-module	0.717/0.543	0.799/0.851	0.808/1.082	0.780/1.270	0.739/1.428	0.691/1.572	0.629/1.749
	Embeddings	0.874/0.896	0.920/1.080	0.925/1.234	0.907/1.366	0.877/1.486	0.841/1.599	0.795/1.721
Epoch 1	L-module	0.782/0.566	0.893/0.757	0.922/0.864	0.918/0.931	0.904/0.968	0.888/0.991	0.870/1.011
	Embeddings	0.899/0.813	0.948/0.894	0.960/0.949	0.954/0.989	0.942/1.016	0.920/1.046	0.908/1.059
Epoch 2	L-module	0.749/0.558	0.846/0.769	0.855/0.886	0.834/0.963	0.767/1.097	0.727/1.268	0.711/1.321
	Embeddings	0.892/0.828	0.934/0.922	0.932/0.991	0.909/1.046	0.841/1.175	0.799/1.331	0.787/1.380
Epoch 3	L-module	0.733/0.603	0.812/0.853	0.735/1.120	0.736/1.169	0.697/1.276	0.636/1.418	0.524/1.763
	Embeddings	0.897/0.869	0.925/0.974	0.892/1.074	0.861/1.143	0.818/1.220	0.782/1.281	0.718/1.401

Table 27: Complete standard looped TF gradient sweep against full $K = 4$ finite backpropagation for the shared Transformer block and embeddings. Each cell reports cosine similarity c and norm ratio r between the K-window and full gradients; $K = 2$ is the native training window.

State	Group	$K = 1$	$K = 2$	$K = 3$	$K = 4$
Scratch	Shared block	0.520/0.323	0.685/0.533	0.829/0.737	1.000/1.000
	Embeddings	0.642/0.291	0.790/0.535	0.913/0.763	1.000/1.000
Epoch 1	Shared block	0.816/0.689	0.930/0.878	0.959/0.940	1.000/1.000
	Embeddings	0.780/0.594	0.928/0.830	0.960/0.926	1.000/1.000
Epoch 2	Shared block	0.733/0.615	0.833/0.784	0.930/0.924	1.000/1.000
	Embeddings	0.714/0.535	0.834/0.743	0.897/0.884	1.000/1.000
Epoch 3	Shared block	0.509/0.619	0.848/0.822	0.922/0.927	1.000/1.000
	Embeddings	0.599/0.577	0.746/0.757	0.881/0.867	1.000/1.000

Table 28: Complete standard looped TF gradient sweep against the IFT-style local reference. Each cell reports cosine c / norm ratio r, averaged over ten matched batches.

State	Group	$K=1$	$K=2$	$K=3$	$K=4$
Scratch	Shared block	0.774/1.254	0.695/2.068	0.563/2.858	0.427/3.878
	Embeddings	0.309/1.817	0.381/3.337	0.352/4.755	0.300/6.232
Epoch 1	Shared block	0.305/0.530	0.450/0.676	0.452/0.723	0.436/0.770
	Embeddings	0.330/0.487	0.479/0.681	0.483/0.760	0.463/0.820
Epoch 2	Shared block	0.333/0.566	0.462/0.722	0.410/0.852	0.395/0.922
	Embeddings	0.359/0.529	0.484/0.735	0.466/0.874	0.422/0.989
Epoch 3	Shared block	0.642/0.645	0.537/0.858	0.540/0.968	0.504/1.044
	Embeddings	0.608/0.617	0.589/0.809	0.617/0.927	0.544/1.069

Increasing the finite window generally improves agreement with full backpropagation. Against the local equilibrium reference, the best window need not be the longest (e.g., $K=3$ or $K=4$ for L^7H), but every native-window recurrent-module and embedding cosine remains positive after training. Thus the aligned reporting recovers the same qualitative result for HRM-Text, L^7H, and the standard looped Transformer despite their different executed and backward depths (eight module calls for HRM-Text and L^7H, and four for the standard looped Transformer).

Takeaway: After training, the truncated gradient stays positively aligned with exact backprop in all three architectures (cosine 0.51–0.96, ten batches each). Its alignment with the IFT-style reference is strong for HRM-Text and L^7H (0.70–0.95) but weaker for the looped TF (0.45–0.59).

D.3 Gradient Derivations for L^7H and the Standard Looped Transformer

D.3.1 Equilibrium Gradients

Both L^7H and the standard looped Transformer contain one shared recurrent map. Their IFT-style gradients therefore require one equilibrium adjoint rather than the coupled Stage 1 and Stage 2 adjoints of HRM-Text. We derive the two cases separately because their output paths differ.

L^7H. To construct the local reference for L^7H, we idealize its repeated L-module phase as

$$s^\star = L(s^\star + x_0; \theta_L), \qquad y = H(s^\star + x_0; \theta_H). \tag{40}$$

At $a^\star = s^\star + x_0$, the local Jacobians and equilibrium response are

$$J_L^{\text{in},\star} = D_a L(a^\star; \theta_L), \qquad J_L^{\theta,\star} = D_{\theta_L} L(a^\star; \theta_L),$$
$$T_L^{\text{IFT}} = (I - J_L^{\text{in},\star})^{-1}, \qquad S_L^{\text{IFT}} = T_L^{\text{IFT}} J_L^{\theta,\star}.$$

Differentiating the equilibrium equation gives

$$(I - J_L^{\text{in},\star})\, ds^\star = J_L^{\text{in},\star}\, dx_0 + J_L^{\theta,\star}\, d\theta_L,$$
$$da^\star = ds^\star + dx_0 = T_L^{\text{IFT}}\, dx_0 + S_L^{\text{IFT}}\, d\theta_L. \tag{41}$$

The second equality uses $I + T_L^{\text{IFT}} J_L^{\text{in},\star} = T_L^{\text{IFT}}$. Let $J_H^{\text{in},\star}$ and $J_H^{\theta,\star}$ denote the input and parameter Jacobians of H at $a^\star$. The output cotangent $q_y = \nabla_y \mathcal{L}$ reaches the recurrent phase as $q_a = (J_H^{\text{in},\star})^\top q_y$. Rather than forming T_L^{IFT}, we obtain $q_x^{\text{IFT}} = (T_L^{\text{IFT}})^\top q_a$ from

$$\big(I - (J_L^{\text{in},\star})^\top\big) q_x^{\text{IFT}} = q_a. \tag{42}$$

With $J_E^\theta = D_{\theta_E} x_0$, the resulting gradients are

$$\nabla_{\theta_H}^{\text{IFT}} \mathcal{L} = \big(J_H^{\theta,\star}\big)^\top q_y, \qquad \nabla_{\theta_L}^{\text{IFT}} \mathcal{L} = \big(S_L^{\text{IFT}}\big)^\top q_a,$$
$$\nabla_{\theta_E}^{\text{IFT}} \mathcal{L} = \big(J_E^\theta\big)^\top q_x^{\text{IFT}}. \tag{43}$$

Thus the final H-module gradient remains a direct derivative, while the gradients of the shared L-module and the embeddings pass through the same recurrent adjoint.

Standard Looped Transformer. For the standard looped Transformer, the same construction applies directly to its repeated map:

$$s^\star = F(s^\star + x_0; \theta_F), \qquad \mathcal{L} = \ell(s^\star; \theta_O). \tag{44}$$

Its local Jacobians and IFT-style responses are

$$J_F^{\mathrm{in},\star} = D_a F(s^\star + x_0; \theta_F), \qquad J_F^{\theta,\star} = D_{\theta_F} F(s^\star + x_0; \theta_F),$$
$$T_F^{\mathrm{IFT}} = (I - J_F^{\mathrm{in},\star})^{-1}, \qquad S_F^{\mathrm{IFT}} = T_F^{\mathrm{IFT}} J_F^{\theta,\star}.$$

The loss cotangent $q_s = \nabla_{s^\star}\mathcal{L}$ therefore requires the single adjoint system for $q_a^{\mathrm{IFT}} = (T_F^{\mathrm{IFT}})^\top q_s$:

$$\big(I - (J_F^{\mathrm{in},\star})^\top\big) q_a^{\mathrm{IFT}} = q_s. \tag{45}$$

The IFT-style gradients are then

$$\nabla_{\theta_F}^{\mathrm{IFT}}\mathcal{L} = \big(S_F^{\mathrm{IFT}}\big)^\top q_s,$$
$$\nabla_{\theta_E}^{\mathrm{IFT}}\mathcal{L} = \big(J_E^\theta\big)^\top \big(q_a^{\mathrm{IFT}} - q_s\big). \tag{46}$$

Unlike L^7H, the output head reads $s^\star$ rather than $s^\star + x_0$. Its embedding cotangent therefore carries the recurrent factor $T_F^{\mathrm{IFT}} - I$ rather than an identity path through a terminal H-module.

With these equilibrium references defined, we next derive the actual truncated gradients used in training, which Appendix D.2.2 compares against these references numerically.

D.3.2 Truncated Gradients for L^7H and the Standard Looped Transformer

The simplified L^7H model and the standard looped Transformer are both instances of a single stopped-tail calculation. Their repeated phase can be written as

$$s_i = M(s_{i-1} + x_0; \theta_M), \qquad i = 1, \dots, N, \tag{47}$$

with per-call input and parameter Jacobians

$$J_{M,i}^{\mathrm{in}} = D_u M(u; \theta_M)|_{u = s_{i-1} + x_0}, \qquad J_{M,i}^\theta = D_{\theta_M} M(s_{i-1} + x_0; \theta_M).$$

The loss cotangent q_N arrives at s_N after any final module. If the stopped graph retains repeated calls $m+1, \dots, N$, exact reverse-mode differentiation gives

$$\nabla_{\theta_M}^{(m+1:N)}\mathcal{L} = \sum_{i=m+1}^{N} \big(J_{M,i}^\theta\big)^\top q_i, \qquad q_i = \big(J_{M,i+1}^{\mathrm{in}}\big)^\top \cdots \big(J_{M,N}^{\mathrm{in}}\big)^\top q_N, \qquad i < N, \tag{48}$$

with q_N used directly when $i = N$. Each retained call also creates a direct gradient path to the repeatedly injected embedding x_0, which is why input injection keeps the embeddings trained even when the earliest recurrent calls are detached from the backward graph.

For L^7H, (2) has $N = 7$, followed by one final H-module call. Writing $q_y = \nabla_y\mathcal{L}$ and $q_7 = (J_H^{\mathrm{in}})^\top q_y$, the native $K = 5$ graph retains $L_4, \dots, L_7, H$ and computes

$$\begin{aligned}
\nabla_{\theta_H}^{(5)}\mathcal{L} &= \big(J_H^\theta\big)^\top q_y, \\
\nabla_{\theta_L}^{(5)}\mathcal{L} &= \big(J_{L,7}^\theta\big)^\top q_7 + \big(J_{L,6}^\theta\big)^\top \big(J_{L,7}^{\mathrm{in}}\big)^\top q_7 \\
&\quad + \big(J_{L,5}^\theta\big)^\top \big(J_{L,6}^{\mathrm{in}}\big)^\top \big(J_{L,7}^{\mathrm{in}}\big)^\top q_7 \\
&\quad + \big(J_{L,4}^\theta\big)^\top \big(J_{L,5}^{\mathrm{in}}\big)^\top \big(J_{L,6}^{\mathrm{in}}\big)^\top \big(J_{L,7}^{\mathrm{in}}\big)^\top q_7.
\end{aligned} \tag{49}$$

Full $K = 8$ backpropagation adds the analogous direct parameter contributions from L_1, L_2, L_3. Because the sole H-module call is retained in every tested window, its parameter gradient and the LM-head gradient are identical to the full finite reference.

For the standard looped Transformer, $N = 4$, there is no parameter-distinct terminal module, and $q_4 = \nabla_{s_4}\mathcal{L}$. Its native $K = 2$ graph retains F_3, F_4:

$$\nabla_{\theta_F}^{(2)}\mathcal{L} = \big(J_{F,4}^\theta\big)^\top q_4 + \big(J_{F,3}^\theta\big)^\top \big(J_{F,4}^{\mathrm{in}}\big)^\top q_4. \tag{50}$$

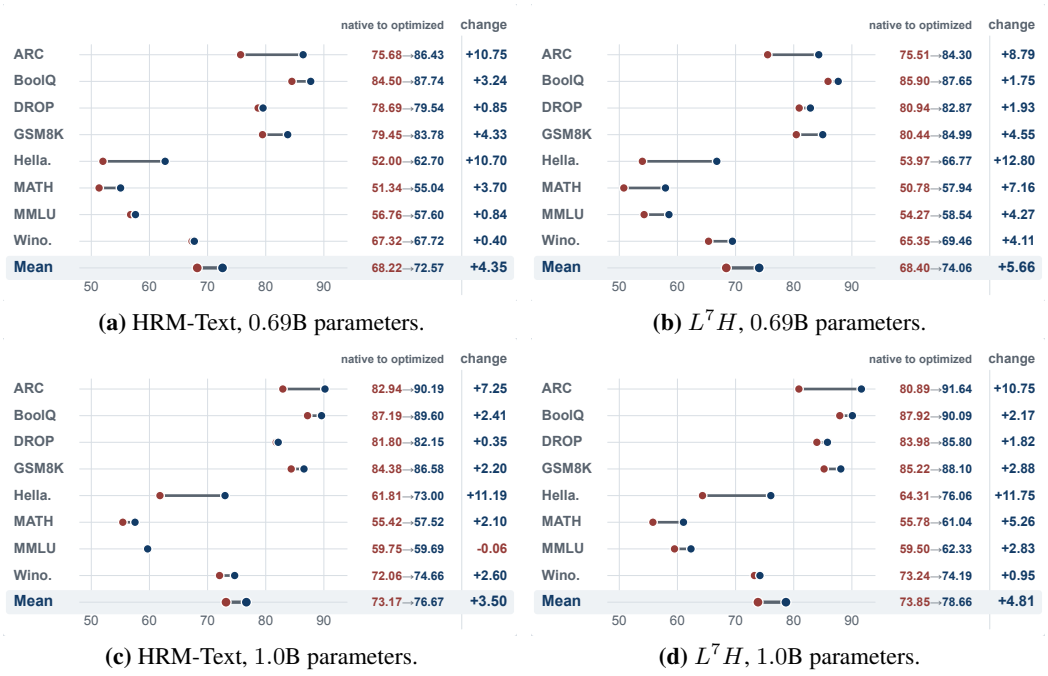
The full four-call reference is

$$\begin{aligned}\nabla^{(4)}_{\theta_F}\mathcal{L} = &\left(J^{\theta}_{F,4}\right)^{\top} q_4 + \left(J^{\theta}_{F,3}\right)^{\top}\left(J^{\mathrm{in}}_{F,4}\right)^{\top} q_4 \\ &+ \left(J^{\theta}_{F,2}\right)^{\top}\left(J^{\mathrm{in}}_{F,3}\right)^{\top}\left(J^{\mathrm{in}}_{F,4}\right)^{\top} q_4 \\ &+ \left(J^{\theta}_{F,1}\right)^{\top}\left(J^{\mathrm{in}}_{F,2}\right)^{\top}\left(J^{\mathrm{in}}_{F,3}\right)^{\top}\left(J^{\mathrm{in}}_{F,4}\right)^{\top} q_4. \end{aligned} \tag{51}$$

Equations (49) and (50) make the common approximation explicit: the numerical forward is unchanged, while earlier direct shared-parameter contributions and the paths leading to them are omitted from backward. The measured native-window agreement is summarized in Table 3 in the main text.

E Optimizer Reproducibility Details

The main text describes the optimizer split and schedule at a high level. Here we record the distributed full-matrix Muon path and the fixed settings needed to reproduce the reported runs; we omit the optimizer sweep history and report only the selected configurations.



(a) HRM-Text, 0.69B parameters. **(b)** L^7H, 0.69B parameters.

(c) HRM-Text, 1.0B parameters. **(d)** L^7H, 1.0B parameters.

Figure 9: Benchmark scores under AdamATan2 and WSD50–Muon at both model scales. Rows connect the original recipe (red) and optimized recipe (navy); right columns report the paired scores and percentage-point changes.

Table 29: Controlled training-recipe progression on 0.69B HRM-Text: constant-schedule AdamATan2, AdamATan2 with WSD50, and WSD50 with the hybrid Muon–AdamATan2 optimizer. Mean is the unweighted average of the eight benchmarks.

Optimizer	Schedule	ARC	BoolQ	DROP	GSM8K	Hella.	MATH	MMLU	Wino.	Mean
AdamATan2	Constant	75.68	84.50	78.69	79.45	52.00	51.34	56.76	67.32	68.22
AdamATan2	WSD50	83.28	**87.83**	**79.85**	81.65	60.45	54.40	57.43	**68.35**	71.65
Muon hybrid	WSD50	**86.43**	87.74	79.54	**83.78**	**62.70**	**55.04**	**57.60**	67.72	**72.57**

Table 30: Exact AdamATan2 and WSD50–Muon scores behind Figure 9; Mean is unweighted.

Model	Params	Recipe	ARC	BoolQ	DROP	GSM8K	Hella.	MATH	MMLU	Wino.	Mean
HRM-Text	0.69B	AdamATan2	75.68	84.50	78.69	79.45	52.00	51.34	56.76	67.32	68.22
HRM-Text	0.69B	WSD50–Muon	**86.43**	**87.74**	**79.54**	**83.78**	**62.70**	**55.04**	**57.60**	**67.72**	**72.57**
L^7H	0.69B	AdamATan2	75.51	85.90	80.94	80.44	53.97	50.78	54.27	65.35	68.40
L^7H	0.69B	WSD50–Muon	**84.30**	**87.65**	**82.87**	**84.99**	**66.77**	**57.94**	**58.54**	**69.46**	**74.06**
HRM-Text	1.0B	AdamATan2	82.94	87.19	81.80	84.38	61.81	55.42	**59.75**	72.06	73.17
HRM-Text	1.0B	WSD50–Muon	**90.19**	**89.60**	**82.15**	**86.58**	**73.00**	**57.52**	59.69	**74.66**	**76.67**
L^7H	1.0B	AdamATan2	80.89	87.92	83.98	85.22	64.31	55.78	59.50	73.24	73.85
L^7H	1.0B	WSD50–Muon	**91.64**	**90.09**	**85.80**	**88.10**	**76.06**	**61.04**	**62.33**	**74.19**	**78.66**

FSDP2 Muon Path. Our distributed core adapts the open-source `muon_fsdp_2` implementation (Jaghouar, 2025). FSDP2, the second version of PyTorch Fully Sharded Data Parallel (FSDP; Zhao et al., 2023), stores each selected matrix and its momentum in row-wise shards, so applying Newton–Schulz independently on each rank would not reproduce a full-matrix Muon update for the four Muon-routed projections in each Transformer layer. We instead gather each momentum matrix to a rotating owner rank, transform the full matrix, and scatter the corresponding shards back. Successive gathers are asynchronous and pipelined across a bounded work queue.

Table 31: Hybrid FSDP2 optimizer; only Muon-routed projections use the full-matrix path.

Phase	Implementation
Sharded momentum	Each rank updates its local momentum shard and forms the Nesterov direction with momentum 0.95.
Matrix reconstruction	Each matrix is assigned to an owner rank round-robin. Its row shards are gathered asynchronously and concatenated on that owner.
Muon transform	The owner applies five BF16 Newton–Schulz iterations to the complete matrix.
Distributed update	The transformed direction is split and scattered back. Each rank applies spectral scaling, decoupled weight decay, and its local update; the AdamATan2 group requires no full-matrix reconstruction.

Fixed Optimizer Settings. AdamATan2 uses $(\beta_1, \beta_2) = (0.9, 0.95)$ and weight decay 0.1. Muon uses Nesterov momentum 0.95, five BF16 Newton–Schulz iterations, and the full-matrix spectral scaling from the cited FSDP2 implementation. The AdamATan2 base learning rate is 2.5×10^{-4} for the 0.69B models and 2.2×10^{-4} for the 1.0B models. The selected Muon learning-rate/weight-decay pairs are $0.003/0.00833$ for 0.69B HRM-Text, $0.002/0.0125$ for 0.69B L^7H, $0.003/0.00833$ for 1.0B HRM-Text, and $0.002/0.0125$ for 1.0B L^7H. The 1.0B L^7H model uses 16-layer L- and H-modules (the XL size, denoted P16/Q16 in our configurations), matching the module depth of the 1.0B HRM-Text model and hence its 128-layer executed depth.

Schedule. All runs use 2,000 linear warmup steps. With total training length T and warmup length $w = 2{,}000$, the WSD50 multiplier is

$$s_t = \begin{cases} t/w, & 0 \le t < w, \\ 1, & w \le t \le T/2, \\ 2(1 - t/T), & T/2 < t \le T. \end{cases} \tag{52}$$

The same multiplier rescales the AdamATan2 and Muon learning rates. The constant-schedule baseline instead holds $s_t = 1$ after warmup. All other settings, including precision, data and batch construction, EMA, and checkpoint selection, follow the HRM-Text baseline unchanged.

Takeaway: Most of the recipe gain comes from the schedule: for 0.69B HRM-Text, WSD50 alone adds 3.43 points (68.22 to 71.65), and the Muon hybrid adds another 0.92 (to 72.57). The combined recipe improves all four model–scale pairs by 3.50–5.66 points.

F Controlled Training-Efficiency Analyses

All profiles use compiled BF16 Fully Sharded Data Parallel (FSDP) (Zhao et al., 2023) on one node with eight H100 GPUs. Reported medians exclude compilation and timing warm-up iterations.

The production measurements combine several sources of systems cost. We separate them using four depths:

$$T_{\text{step}} \approx C_f(D_f) + C_p(D_p) + C_b(D_g) + C_r(D_r), \tag{53}$$

where D_p, D_f, D_g, D_r denote trainable depth, executed depth, retained-gradient depth, and checkpoint-recomputation depth, as in the main text. The last quantity is a memory-dependent policy rather than an architecture constant: increasing the per-GPU batch or D_g retains more activations and can force $D_r > 0$ under a fixed memory budget. Recomputation then lowers memory at the cost of additional forward work. All controlled profiles fix $D_f = 96$ and use AdamATan2. Sharing reduces D_p and permits $D_g < D_f$ while still updating every reused layer; applying the same truncation to an unshared model would leave its unique prefix layers untrained.

The modular standard Transformer is the eight-independent-module control from Section 2.3; the monolithic control implements the same 96 distinct and executed layers as one stack. Their similar profiles check that the comparison is not an artifact of module layout.

Production Operating Point. The native comparison uses an effective global batch of 172,032 tokens, or 21,504 tokens per GPU with one accumulation step. At this operating point, the recurrent models' shorter retained-gradient depths fit without checkpointing, while both full-gradient standard Transformers require $D_r = 48$. Table 32 reports these policies and their time and memory costs.

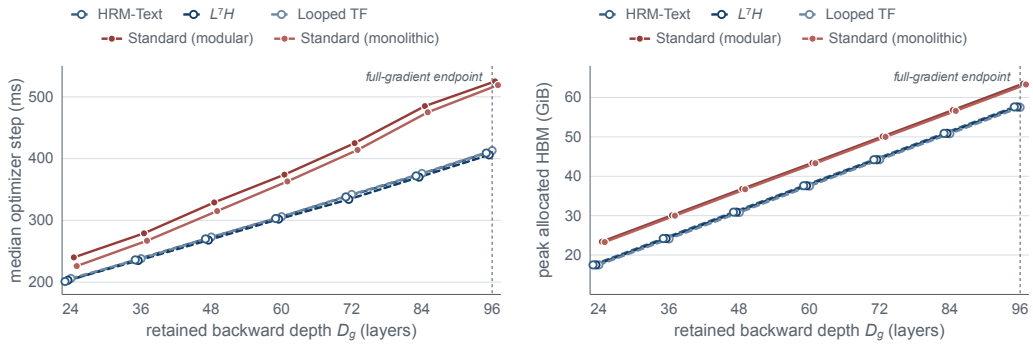
Takeaway: At the production batch, the three recurrent models fit in memory without recomputation and process 60B tokens in 48.7–52.5 hours, versus 106.2–107.6 hours for the 96-layer standard Transformers, which must checkpoint and recompute 48 of their 96 layers.

Table 32: Production-policy efficiency at a common executed depth of 96 layers and a 172,032-token effective global batch. D_p, D_g, and D_r are the trainable, retained-gradient (over the $K = 2 \to 5$ warmup), and checkpoint-recomputation depths in Transformer layers. Step is the median step time, ktok/s is throughput, HBM is peak allocated memory, and 60B (h) is hours per 60B tokens.

Model	Recipe	D_p	D_g	D_r	Step (ms)	ktok/s	HBM (GiB)	60B (h)
HRM-Text	AdamATan2	24	24–60	0	540	318.6	72.3	52.3
L^7H	AdamATan2	24	24–60	0	542	317.4	72.3	52.5
Standard Looped TF	AdamATan2	24	48	0	503	342.0	58.8	48.7
Standard Transformer (modular)	AdamATan2	96	96	48	1,111	154.9	67.6	107.6
Standard Transformer (monolithic)	AdamATan2	96	96	48	1,096	156.9	64.9	106.2
HRM-Text	WSD50–Muon	24	24–60	0	606	284.1	72.1	58.7
L^7H	WSD50–Muon	24	24–60	0	606	283.8	72.1	58.7

F.1 Checkpoint-Free Backward-Depth Sweep

The first control isolates the cost of retaining additional gradient depth without changing the checkpoint policy. We halve the per-GPU batch from the production value of 21,504 to 10,752 tokens, giving an effective global batch of 86,016 tokens with one accumulation step. This smaller batch allows every profile to reach $D_g = 96$ with $D_r = 0$. Figure 10 then sweeps D_g under this common checkpoint-free setting. Every shared-parameter layer receives gradients throughout the recurrent-model sweep. The standard-Transformer points with $D_g < 96$ remain systems diagnostics because they omit gradients for the unique prefix layers of an unshared model.



(a) Retained depth determines optimizer-step time. **(b)** Retained depth determines activation memory.

Figure 10: Controlled retained-gradient-depth sweep for five architectures at $D_f = 96$, $D_r = 0$, and an 86,016-token effective global batch. (a) Compiled optimizer-step time and (b) peak allocated HBM as D_g increases. Curves are offset slightly to reveal overlaps; lower is better.

Increasing D_g raises both step time and activation memory. At the matched full-gradient endpoint $D_g = 96$, the remaining separation between recurrent and standard models reflects parameter count, communication, and optimizer work associated with sharing, without any recomputation.

Takeaway: With executed depth fixed at $D_f = 96$ and no recomputation, increasing retained-gradient depth D_g raises both step time and activation memory; at matched D_g, the remaining differences are smaller and include parameter count, communication, and optimizer work.

F.2 Checkpoint-Free Batch Scaling at Matched Depth

The second control asks how the architectures scale with batch size after matching the backward graph. Fixing $D_g = 48$ allows all five profiles to remain at $D_r = 0$ as the effective global batch reaches the production value of 172,032 tokens. Figure 11 therefore varies batch size while holding both retained depth and checkpoint recomputation fixed. As above, the standard-Transformer rows are systems controls rather than faithful training policies because their unique prefix layers lie outside the retained gradient tail and would never be updated during training.

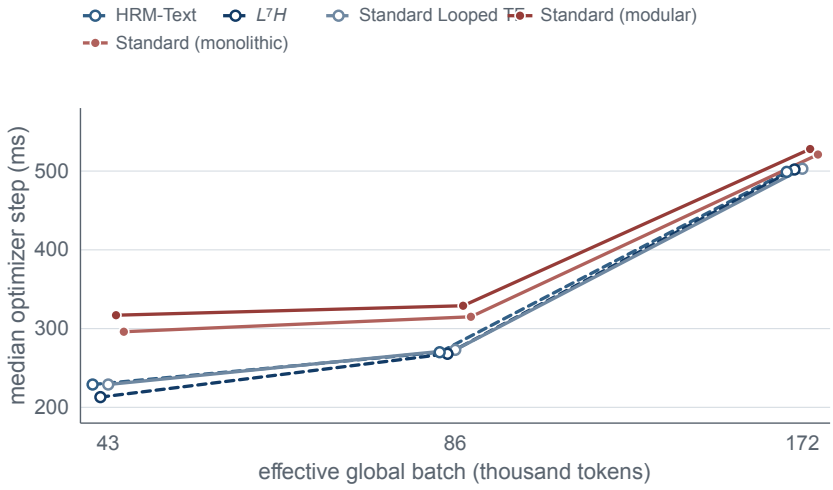


Figure 11: Controlled optimizer-step time for five architectures as effective global batch size increases, with $D_f = 96$, $D_g = 48$, and $D_r = 0$ held fixed. Curves are slightly offset; lower is better.

The shared models remain in a lower-latency band at matched D_g and D_r, but the gap is much smaller than under the native production policies. The two controls explain the production result: shorter recurrent backward graphs fit in memory without checkpointing, whereas full-gradient standard Transformers cross that limit and must recompute 48 layers. The production latency advantage thus comes from the shorter retained-gradient depth and, at this batch, from avoiding that recomputation; parameter sharing accounts for the smaller residual difference under matched backward policies.

Takeaway: At this batch, the speedup comes from the shorter retained-gradient depth, which sharing enables without leaving layers untrained, and from avoiding the 48-layer recomputation of full-gradient standard Transformers; sharing adds a smaller gain under matched backward policies.

G Data, Evaluation, and Transfer Analyses

G.1 Training-Mixture Composition and Source Overlap

Table 33: HRM-Text training-mixture components and core-benchmark training-set overlap, reconstructed from Wang et al. (2026) and the released data pipeline. CBO denotes documented use of a core benchmark's training split; an em dash indicates that neither source documents any direct or constituent-source overlap for that component with any core benchmark.

Source group	Representative datasets	Coverage and CBO
General instructions	FLAN (Longpre et al., 2023); Tasksource (Sileo, 2024); NoRobots (Rajani et al., 2023)	Instructions and broad task execution. *CBO via FLAN:* ARC-C, BoolQ, DROP, GSM8K, HellaSwag, and WinoGrande.
Rewritten knowledge	SYNTH (PleIAs, 2025)	Factual and encyclopedic knowledge. *CBO:* —
Math and reasoning	GSM8K (Cobbe et al., 2021); MATH (Hendrycks et al., 2021b); Platypus (Lee et al., 2023); Principia (Aggarwal et al., 2026); OpenMathInstruct2 (Toshniwal et al., 2025); NuminaMath (Li et al., 2024); OmniMATH (Gao et al., 2025)	Mathematical problem solving. *CBO:* GSM8K and MATH, directly and through OpenMathInstruct2 and NuminaMath.
Symbolic	DMMath (Saxton et al., 2019); AMPS (Hendrycks et al., 2021b); Sudoku-Extreme (Wang et al., 2025)	Formal mathematics and combinatorial solving. *CBO:* —
Answer-only reasoning	AceReason (Chen et al., 2025); OpenThoughts2 (Guha et al., 2026)	Reasoning with explicit thinking spans removed. *CBO:* GSM8K and MATH through constituent math corpora.
Textbook exercises	TextbookReasoning (Fan et al., 2026a)	Academic and textbook problems. *CBO:* —
Web instructions	NaturalReasoning (Yuan et al., 2025); WebInstruct-verified (Ma et al., 2025); AMPS-khan (Hendrycks et al., 2021b)	Diverse web-derived questions. *CBO:* —

Table 34: Comparison among models trained in this study on seven source-matched held-out evaluations: DROP (Dua et al., 2019), GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), OpenBookQA (OBQA) (Mihaylov et al., 2018), ReClor (Yu et al., 2020), ScienceQA (SciQA) (Lu et al., 2022), and DeepMind Mathematics (DMMath) (Saxton et al., 2019). Mean is the unweighted average of the seven displayed metrics; the best value per scale is in bold.

Model	Params	Recipe	DROP	GSM8K	MATH	OBQA	ReClor	SciQA	DMMath	Mean
Standard Looped TF	0.69B	AdamATan2	70.0	67.6	43.6	65.4	67.6	87.7	79.0	68.7
Standard Looped TF + H	1.22B	AdamATan2	76.0	77.1	50.3	70.4	**88.8**	92.9	80.3	76.5
HRM-Text	0.69B	AdamATan2	78.7	79.5	51.3	73.0	75.8	**93.3**	79.6	75.9
HRM-Text	0.69B	WSD50–Muon	79.5	83.8	55.0	**76.6**	87.0	90.3	83.9	79.4
L^7H	0.69B	AdamATan2	80.9	80.4	50.8	71.4	81.6	92.9	78.9	76.7
L^7H	0.69B	WSD50–Muon	**82.9**	**85.0**	**57.9**	73.8	83.8	91.1	**84.1**	**79.8**
HRM-Text	1.0B	AdamATan2	81.8	84.4	55.4	77.2	84.2	94.5	83.7	80.2
HRM-Text	1.0B	WSD50–Muon	82.2	86.6	57.5	79.2	**92.2**	90.6	**85.3**	81.9
L^7H	1.0B	AdamATan2	84.0	85.2	55.8	76.8	86.2	**94.8**	80.9	80.5
L^7H	1.0B	WSD50–Muon	**85.8**	**88.1**	**61.0**	**80.6**	90.0	94.2	**85.3**	**83.6**

G.2 Evaluation Protocols for Published Comparison Models

Table 35 records how the published-model entries in Table 5 were obtained. We use an official score whenever one is available for the target benchmark. When a related target has not been reported,

we first reproduce the model's score on the reported source benchmark using its disclosed recipe, then transfer that recipe to the related target. GSM8K provides this anchor for GSM-Symbolic, and MMLU or MMLU-Pro can analogously anchor the other member of that pair. If both benchmarks in a pair have official results, we retain both instead of reevaluating them. A compatible reconstruction is used only when no reproducible public protocol is available. OLMES denotes the standardized evaluation recipes of Gu et al. (2025). These published-model rows provide capability references rather than controlled architecture comparisons, since the models differ in data, scale, and training. Local reproductions can differ slightly from the HRM-Text-reported values in Table 4.

Table 35: Official scores used and local evaluation recipes for the published comparison models in Table 5. A transferred recipe is first checked on its reported source benchmark and only then applied to the related target benchmark (e.g., from GSM8K to GSM-Symbolic).

Model	Official scores used	Local evaluation and transfer recipe
Olmo 3 7B Base	BBH (source-native 3-shot)	GSM8K: exact OLMES 8-shot CoT reproduction, temperature 0.6, top-p 0.6, and eight independently scored draws; GSM-Symbolic: transfer of this validated recipe. MMLU-Pro: exact 5-shot OLMES candidate-likelihood reproduction; MMLU: transfer. IFEval: raw prompting of our design with greedy decoding.
Gemma 3 4B Base	MMLU; MMLU-Pro; BBH	GSM8K: exact public OLMES 8-shot sampled-CoT reproduction; GSM-Symbolic: transfer of this validated recipe. IFEval: raw prompting of our design with greedy decoding.
Qwen3 8B Base	MMLU; MMLU-Pro; BBH	GSM8K: exact public 4-shot CoT reproduction with greedy decoding and the published answer-cleaning parser; GSM-Symbolic: transfer of this validated recipe. IFEval: raw prompting of our design with greedy decoding.
Ouro 1.4B Base	MMLU; MMLU-Pro; BBH	GSM8K: source-native 3-shot CoT reproduction with greedy decoding; GSM-Symbolic: transfer of this validated recipe. IFEval: raw prompting of our design with greedy decoding.
Olmo 3 7B Instruct-SFT / Think-SFT	BBH	GSM8K: exact zero-shot OLMES chat reproduction, temperature 0.6, top-p 0.95, and three independently scored draws; GSM-Symbolic: transfer of this validated recipe. MMLU: exact zero-shot OLMES CoT reproduction; MMLU-Pro: transfer. IFEval: exact zero-shot OLMES recipe and official scorer; Think-SFT removes the thinking trace before scoring.
Gemma 3 4B IT	MMLU; MMLU-Pro; BBH; IFEval	GSM8K: compatible zero-shot CoT reconstruction with greedy decoding, answer-aware scoring, and an 8,192-token output cap; GSM-Symbolic: transfer of the same recipe.

For locally evaluated IFEval rows, both columns report instruction-level loose accuracy over 834 instructions from the official 541-prompt evaluator with a 3,072-token output cap. The three-shot condition adds fixed handwritten demonstrations to the corresponding zero-shot prompt. Base models use raw prompts, whereas the Olmo SFT models follow the public OLMES chat recipe and have no comparable three-shot result. For sampled recipes, each generation is scored independently rather than by majority vote, so the reported score is the mean over samples.

G.3 Sudoku-Extreme Analysis

Sudoku-Extreme is explicitly included in the training mixture and permits a granular symbolic-task analysis: its outputs have a fixed structure and distinguish supplied clues from cells the model must infer. A correct solution must produce a valid 81-digit grid, preserve the clues, and infer the missing cells correctly. We therefore report exact accuracy together with output validity, clue-copying accuracy, and non-clue accuracy, which isolates the cells the model must infer.

Evaluation Protocol. We evaluate generated solutions on the held-out test split used by HRM (Wang et al., 2025) and HRM-Text (Wang et al., 2026) and on a size-matched training-split sample. Table 36 reports exact accuracy on both splits and detailed test diagnostics. "81-char" counts outputs of the requested length, while "malformed" marks outputs containing whitespace or non-digit characters. Position-wise accuracy credits legal digits that match the reference; clue and non-clue accuracy distinguish copying supplied cells from solving missing ones.

Table 36: Sudoku-Extreme exact accuracy and output diagnostics. The Train column reports exact accuracy on the sampled training split; all remaining diagnostics use the test split. Position-wise metrics retain partial credit for legal digits in malformed 81-character outputs. Arrows indicate the preferred direction of each metric (higher or lower is better).

	Exact accuracy		Test output form				Test position-wise accuracy		
Model	Test ↑	Train ↑	Parseable ↑	81-char ↑	Malformed ↓	Illegal/malf. ↓	Cell ↑	Clue ↑	Non-clue ↑
Standard Looped TF, 0.69B	2.58	2.58	99.07	99.90	0.83	1.01	60.32	99.52	42.61
Standard Looped TF + H, 1.22B	2.33	2.32	97.15	99.84	2.69	1.02	59.22	**99.81**	40.89
HRM-Text, 0.69B AdamATan2	2.22	2.19	98.35	**100.00**	1.65	1.02	58.60	98.28	40.67
HRM-Text, 0.69B WSD50–Muon	2.07	2.06	91.98	99.17	7.19	1.08	57.85	98.94	39.29
L^7H, 0.69B AdamATan2	1.91	1.93	**99.56**	99.99	0.43	1.01	58.38	99.58	39.76
L^7H, 0.69B WSD50–Muon	1.94	1.95	70.72	96.78	26.05	1.14	56.38	96.45	38.27
HRM-Text, 1.0B AdamATan2	2.34	2.33	96.27	99.98	3.70	1.08	59.67	99.45	41.70
HRM-Text, 1.0B WSD50–Muon	2.63	2.64	99.42	99.69	**0.27**	1.21	59.26	99.20	41.21
L^7H, 1.0B AdamATan2	**5.19**	**5.16**	98.78	99.38	0.60	**1.00**	**65.27**	99.29	**49.90**
L^7H, 1.0B WSD50–Muon	2.86	2.89	95.13	98.75	3.62	1.04	59.51	98.56	41.87

Across all ten checkpoints, test exact accuracy ranges from 1.91% to 5.19%, although at least 96.78% of outputs contain 81 characters and non-clue accuracy remains below 50%. Train and test exact accuracy differ by at most 0.04 percentage points. The models therefore reproduce the grid structure and supplied clues far more reliably than they infer missing cells; detailed training-split diagnostics appear in Appendix Table 37, computed on the sampled training split.

G.3.1 Detailed Training-Split Diagnostics

Table 37 complements the test diagnostics above using the fixed seed-0 sample of 422,786 rows from the Sudoku-Extreme training split. Exact accuracy and parseability use the strict 81-digit output rule. "81-char" also includes malformed outputs of the requested length, and position-wise metrics credit only legal digits that match the reference at the same position.

Table 37: Detailed diagnostics on the sampled Sudoku-Extreme training split. All values are percentages except Illegal/malf.: mean illegal characters per malformed 81-character output.

	Output and format					Position-wise accuracy		
Model	Exact	Parseable	81-char	Malformed	Illegal/malf.	Cell	Clue	Non-clue
Standard Looped TF, 0.69B	2.58	99.11	99.91	0.80	1.01	60.39	99.53	42.69
Standard Looped TF + H, 1.22B	2.32	97.14	99.83	2.69	1.02	59.30	99.80	40.99
HRM-Text, 0.69B AdamATan2	2.19	98.32	100.00	1.68	1.02	58.66	98.29	40.74
HRM-Text, 0.69B WSD50–Muon	2.06	91.96	99.20	7.24	1.09	57.95	98.97	39.40
L^7H, 0.69B AdamATan2	1.93	99.58	99.99	0.41	1.00	58.45	99.58	39.85
L^7H, 0.69B WSD50–Muon	1.95	70.79	96.76	25.97	1.14	56.47	96.43	38.41
HRM-Text, 1.0B AdamATan2	2.33	96.29	99.98	3.69	1.09	59.72	99.45	41.77
HRM-Text, 1.0B WSD50–Muon	2.64	99.42	99.68	0.26	1.22	59.36	99.20	41.35
L^7H, 1.0B AdamATan2	5.16	98.81	99.40	0.59	1.01	65.42	99.31	50.10
L^7H, 1.0B WSD50–Muon	2.89	95.15	98.76	3.62	1.04	59.62	98.58	42.00

The training-split format and position-wise results closely track their test counterparts in Table 36. Thus the absence of a training-split advantage is not limited to exact accuracy: the sampled training rows also show no broad improvement in output conformity, clue copying, or non-clue solving.

G.3.2 Contribution Beyond Sudoku

We also regenerate the training mixture after removing Sudoku before tokenization and resampling the remaining sources. Architecture, optimizer, and evaluation remain unchanged.

Table 38: Matched HRM-Text runs trained with the original mixture or with Sudoku removed before resampling the remaining sources. Difference is No-Sudoku minus Original, and Mean is the unweighted average of the eight core benchmark scores for each run.

Training mixture	ARC	BoolQ	DROP	GSM8K	Hella.	MATH	MMLU	Wino.	Mean
Original	**75.68**	84.50	78.69	79.45	**52.00**	51.34	**56.76**	**67.32**	68.22
Resampled without Sudoku	75.43	**85.93**	**79.20**	**80.59**	51.66	**51.40**	56.31	65.35	**68.23**
Difference	-0.25	+1.43	+0.51	+1.14	-0.34	+0.06	-0.45	-1.97	+0.01

Removing Sudoku changes individual scores in both directions but leaves their mean effectively unchanged. Together, the results distinguish format and clue copying from reliable exact solving and show no aggregate gain on the general benchmark suite under this training setup.

Takeaway: Models trained with Sudoku reproduce its format and clues but rarely solve it (clue accuracy at least 96%, non-clue accuracy below 50%, exact accuracy at most 5.19%), and removing it leaves the eight-benchmark mean unchanged (68.23 vs. 68.22 for the original mixture).

G.4 Safety-Transfer Results and SFT Configuration

We fine-tune the 0.69B HRM-Text checkpoints; Table 39 reports all three safety-data variants and all four pretraining-to-SFT optimizer transitions, which fine-tune on WildJailbreak (Jiang et al., 2024), WildGuardMix (Han et al., 2024), or their equal mixture, following Tülu 3 (Lambert et al., 2025). Utility is the unweighted mean of the eight general benchmarks, and safety is the direct-request HarmBench score (Mazeika et al., 2024). Changes are relative to the pre-SFT source checkpoint.

Table 39: Complete safety-transfer grid across source checkpoints, SFT optimizers, and safety-data mixtures. Utility is the unweighted mean of the eight general benchmarks, safety is direct-request HarmBench score, and both Δ columns are measured relative to the matching source checkpoint. WJ denotes WildJailbreak, WGM denotes WildGuardMix, and Combined contains equal portions of both sources (Appendix Table 40 lists the corresponding SFT optimizer settings).

SFT optimizer	Data	Utility	Δ utility	Safety	Δ safety
AdamATan2-pretrained 0.69B HRM-Text: utility 68.22, safety 24.69					
AdamATan2	WJ	68.10	-0.12	97.50	+72.81
AdamATan2	WGM	68.20	-0.02	95.31	+70.62
AdamATan2	Combined	**68.47**	**+0.25**	**99.38**	**+74.69**
Muon hybrid	WJ	68.25	+0.03	96.56	+71.87
Muon hybrid	WGM	68.10	-0.12	90.00	+65.31
Muon hybrid	Combined	68.32	+0.10	94.06	+69.37
WSD50–Muon-pretrained 0.69B HRM-Text: utility 72.57, safety 29.69					
AdamATan2	WJ	69.94	-2.63	96.88	+67.19
AdamATan2	WGM	71.01	-1.56	93.13	+63.44
AdamATan2	Combined	70.54	-2.03	95.00	+65.31
Muon hybrid	WJ	71.09	-1.48	**98.44**	**+68.75**
Muon hybrid	WGM	71.16	-1.41	94.69	+65.00
Muon hybrid	Combined	**71.20**	**-1.37**	96.88	+67.19

The three data arms use deterministic training/validation splits. WJ contains 47,500 training and 2,500 validation examples, WGM contains 46,775 and 2,500, and Combined contains 23,750 training examples plus 1,250 validation examples from each source. HarmBench evaluation uses 320 prompts, deterministic generation with at most 512 new tokens, and the WildGuard classifier.

Table 40: SFT optimizer configurations. η_M, η_A and λ_M, λ_A denote learning rates and weight decays for the spectral-Muon and AdamATan2 groups. All runs use three-epoch full-parameter SFT, a 32,768-token global batch, EMA 0.999, no warm-up, and eight-way FSDP on one eight-H100 node. AdamATan2 uses $(\beta_1, \beta_2) = (0.9, 0.95)$; Muon uses Nesterov momentum 0.95 and five Newton–Schulz steps. Final EMA checkpoints are evaluated for both utility and safety.

Transition	SFT optimizer	η_M	η_A	λ_M	λ_A	Schedule
Adam to Adam	AdamATan2	–	0.00004	–	0.1	Cosine, min. ratio 0.1
Adam to Muon	Muon hybrid	0.0003	0.00003	0.0083333	0.1	WSD50
Muon to Adam	AdamATan2	–	0.000006	–	0.1	WSD50
Muon to Muon	Muon hybrid	0.00015	0.0000125	0.0083333	0.1	WSD50

Takeaway: Safety fine-tuning of HRM-Text on unseen data raises direct-request HarmBench safety by 63–75 points in all twelve runs, while general utility changes by -2.63 to $+0.25$ points relative to the source checkpoint; over-refusal on benign requests was not measured.