How Much Is One Recurrence Worth? Iso-Depth Scaling Laws for Looped Language Models

Kristian Schwethelm[1] **Daniel Rückert**[1,2,3] **Georgios Kaissis**[4]

[1]Chair for AI in Healthcare and Medicine, Technical University of Munich (TUM) and TUM University Hospital, Munich, Germany
[2]Department of Computing, Imperial College London, UK
[3]Munich Center for Machine Learning (MCML), Germany
[4]Hasso Plattner Institute for Digital Engineering, University of Potsdam, Germany

Abstract

We measure how much one recurrence is worth to a looped transformer in equivalent unique parameters. From an iso-depth pretraining sweep across recurrence counts $r \in \{1, 2, 4, 8\}$ spanning $\sim 50\times$ in training compute, we fit a joint scaling law $L = E + A\,(N_{\text{once}} + r^{\varphi} N_{\text{rec}})^{-\alpha} + B\,D^{-\beta}$ and measure a *recurrence-equivalence exponent* φ=0.46. Intuitively, φ tells us whether looping a block r times is equivalent in validation loss to r unique blocks of a non-looped model (full equivalence, φ=1) or to a single block run repeatedly with no capacity gain (φ=0). Our φ=0.46 sits in between, so replacing unique blocks with shared recurrences *increases* validation loss at matched training compute. For example, at r=4 a 410M looped model performs on par with a 580M non-looped model, but incurs the training cost of a 1B non-looped one. We demonstrate the utility of φ as a diagnostic tool on two case studies: commonly used truncated backpropagation lowers φ to 0.38, indicating that the loop mechanism is poorly trained under truncation, even though validation loss decreases. Conversely, hyperconnections raise φ to 0.65, a genuine capacity gain. Our method separates true loop improvements from training-side gains, a distinction raw validation loss cannot make. Code is available at https://github.com/kschwethelm/looped-lm-scaling.

1 Introduction

Can a transformer block looped r times replace r non-looped blocks at matched compute budget? Looped transformers iterate a shared block of layers, increasing model depth without adding new layers (parameters) [1, 2]. Such models have recently drawn attention in language modelling as a route to implicit, latent-space reasoning and test-time compute scaling [3–9]. However, in practice, most looped LMs use only small recurrence counts, revealing a cost: applying a block r times costs the same compute (FLOPs) as applying r unique blocks (i.e., a non-looped model of the same depth). A looped model therefore performs the same number of layer computations with fewer unique parameters. To what extent repeated applications of shared parameters can recover the capacity provided by unique blocks remains unclear. We measure this using an iso-depth scaling analysis.

For this, we compare model performance across recurrence counts $r \in \{1, 2, 4, 8\}$, where r=1 is a conventional non-looped transformer. As r increases, we reduce the number of layers in the shared recurrent block so that every architecture performs the same number of layer computations per token, making them *iso-depth* (Figure 2). At matched model width, the compared models then have approximately the same per-token training and inference FLOPs, while their number of unique

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

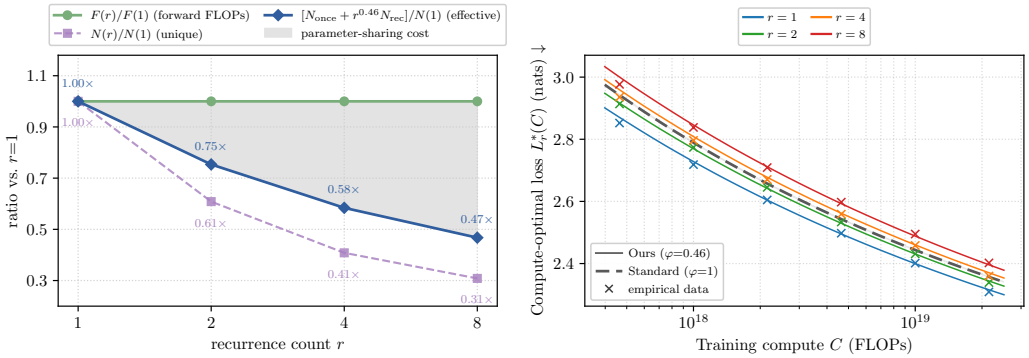


Figure 1: *Left:* at matched effective depth, per-token forward FLOPs $F(r)$ stay flat while unique parameters $N(r)$ drop as r grows. Shared parameter capacity $N_{\text{once}} + r^{\varphi} N_{\text{rec}}$ with φ=0.46 drops more slowly. *Right:* compute-optimal validation loss per architecture against compute budget C. Empirical per-budget optima (crosses) track our φ=0.46 fit (solid curves). The standard form (φ=1) collapses all four architectures onto a single curve (dashed), which fits none of the empirical results.

parameters decreases by a factor of up to 3.2 (Figure 1, left). We evaluate this trade-off with a pretraining scaling law across six compute budgets from 4.64×10^{17} to 2.15×10^{19} FLOPs.

The standard Chinchilla scaling law $L(N, D)$ [10] can relate validation loss to unique parameters N and training tokens D, but has limited utility in comparing looped models. The recurrence count r does not enter the law, and the parameter count N has different FLOPs cost under looping. We therefore introduce a joint scaling law $L(N_{\text{once}}, N_{\text{rec}}, D, r)$ that includes r and splits N into the shared recurrent parameters (N_{rec}) and the single-use ones (N_{once}), with $N_{\text{once}} + N_{\text{rec}} = N$.

At the core of our joint scaling law, we introduce a *recurrence-equivalence exponent* φ that quantifies how much each shared parameter contributes to loss reduction relative to a unique one: $L(N_{\text{once}}, N_{\text{rec}}, D, r) = E + A\,(N_{\text{once}} + r^{\varphi} N_{\text{rec}})^{-\alpha} + B\,D^{-\beta}$. φ has two natural reference points. $\varphi = 1$ attributes full parameter equivalence to each recurrence (no sharing cost), so at matched training FLOPs the looped model would reach the same validation loss as the non-looped baseline. $\varphi = 0$ indicates no gain through recurrences (a pure sharing cost). For our baseline architecture, we find $\varphi = 0.46$ (Figure 1). This means a 410M r=4 model performs on par with a 580M non-looped model, but incurs the training cost of a 1B non-looped one (derivation in Appendix G.5).

Is φ fixed? No. We propose the shift $\Delta\varphi$ as a diagnostic tool and demonstrate it on two case studies: truncated backpropagation (gradients are only computed for the last few loops) [3, 7, 9] and hyperconnections (parallel residual streams between loops) [11]. Both decrease validation loss but move φ in opposite directions. Truncated backpropagation decreases φ to 0.38, so each loop becomes *less* powerful. Hyperconnections raise φ to 0.65. We further show that $\Delta\varphi$ distinguishes whether a validation-loss gain comes from a better loop mechanism or a hidden trade-off with inference cost.

Contributions.

1. We introduce a *recurrence-equivalence exponent* φ that quantifies, via a joint scaling law over r, how many unique parameter blocks one recurrence is worth, and measure a baseline of $\varphi = 0.46$: each recurrence is worth roughly half a unique block at the same FLOPs.

2. We propose $\Delta\varphi$ as a diagnostic tool for comparing looped LM training recipes and architectures. It separates architecture-side from training-side gains, with the compute-optimal allocation reflecting the resulting deployment cost.

3. By measuring $\Delta\varphi$, we show that commonly used truncated backpropagation weakens looping and raises inference cost ($\varphi = 0.38$), while hyperconnections provide a genuine capacity gain that also narrows the compute-optimum ($\varphi = 0.65$).

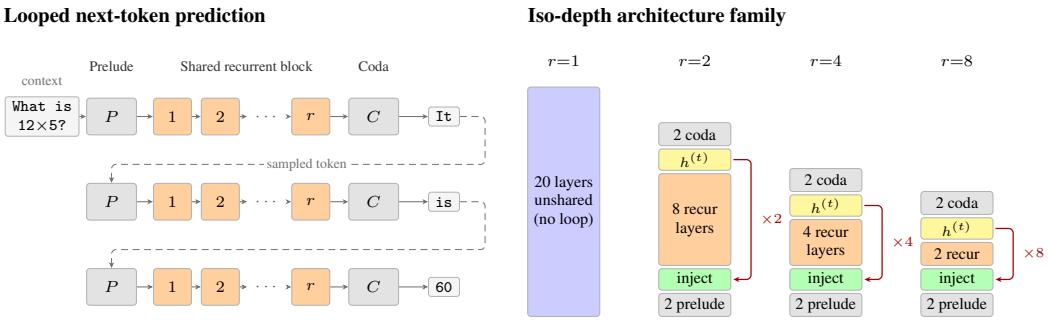


Figure 2: *Left:* each next-token prediction runs the prelude once, the shared recurrent block r times, and the coda once. Input injection between recurrences is omitted for clarity. *Right:* our architectures keep the effective depth fixed, e.g., to 20 by reducing the recurrent block from 8 to 4 to 2 layers as r increases from 2 to 4 to 8. The r=1 is a conventional 20-layer non-looped transformer.

2 Related Work

Looped language models. Huginn [3] and Ouro [4] have scaled looped LMs to ∼3B parameters and trillion-token training budgets with strong downstream results, often matching much larger dense transformers. However, this comes at a proportional training-compute cost, as each loop multiplies forward-backward FLOPs. Compute-matched comparisons to non-looped models at single training budgets report a consistent pattern: looped models trail non-looped baselines on validation loss and parametric-knowledge tasks but close the gap or outperform them on reasoning benchmarks [12]. We extend these findings to the scaling-law setting with various training budgets. Unlike per-architecture scaling laws [10, 13], our joint law fits looped and non-looped models together under a single recurrence-equivalence exponent φ. Architectural and training-efficiency methods such as retrofitting [7, 8], adaptive compute [5, 6, 14, 15], truncated backpropagation [3, 7, 9], and hyperconnections [11, 16] affect φ. We apply our framework to truncated backpropagation and hyperconnections, leaving the others to future work. See Appendix A for extended discussion.

Looped scaling laws. Concurrent work [9] fits a scaling law at fixed parameter count to identify compute optimal recurrence, motivated by keeping weight memory overhead fixed. However, in this setting, per-token inference FLOPs and KV cache memory grow with recurrence compared to the baseline, which can dominate inference cost (Appendix D). In contrast, our iso-depth setup keeps both costs approximately fixed and characterises compute-optimal behaviour under this constraint. We additionally focus on directly measuring the per-recurrence parameter-sharing cost.

3 Methodology

3.1 Looped Transformer Architecture

We compare four transformer variants: a non-looped baseline (r=1) and looped models with $r \in \{2, 4, 8\}$ recurrences. All looped variants follow the prelude-recur-coda architecture [3]. Here, an unshared prelude first embeds the tokens, then the shared recurrent block refines the hidden state $h^{(t)}$ with r loops, and finally the unshared coda predicts the next token (Figure 2). This allows some layers to specialize in input processing and prediction while parameter sharing is confined to the middle of the network. We fix the effective depth at $\ell_{\text{eff}} = \ell_{\text{prelude}} + r \cdot \ell_{\text{recur}} + \ell_{\text{coda}} = 20$. Setting $(\ell_{\text{prelude}}, \ell_{\text{coda}}) = (2, 2)$ gives a recurrent block of $\ell_{\text{recur}} = 16/r$ layers, corresponding to $(8, 4, 2)$ layers for $r \in \{2, 4, 8\}$. Width is parameterised as $d_{\text{model}} = 64s$ for an integer scale factor s, with attention head dimension 128. Following Geiping et al. [3], a linear input-injection layer combines the prelude output with the recurrent state before each recurrence. Its small FLOPs overhead is included in every iso-FLOPs comparison. Full architectural details are provided in Appendix C.

Let n_b=$12d^2$ be the parameter count of a single transformer layer at width d≡d_{model} (four d×d attention projections plus the $d \to 4d \to d$ MLP), and n_i=$2d^2$ that of the injection layer. We write N for the transformer's total non-embedding parameters and adopt the standard convention that per-token forward FLOPs ≈$2N$ and training FLOPs ≈$6N$ [13]. Our reported FLOPs also include

parameter-free attention matmuls ($12\,hqT$ at training, with h heads of dimension q and sequence length T) and the unembedding matmul ($6dV$ at training, with vocabulary V), both fixed across architectures at matched width. For simplicity we exclude them from the equations below.

For looped models, we split N into parameters that are used once, $N_{\text{once}} = (\ell_{\text{prelude}} + \ell_{\text{coda}})\, n_b$, and those that are recurring, $N_{\text{rec}} = \ell_{\text{recur}}\, n_b + n_i$ (fully-looped architectures correspond to $N_{\text{once}} = 0$). In our setup $\ell_{\text{recur}} = 16/r$, so each additional loop shrinks the recurrent block. Thus, our looped models with $r \in \{2, 4, 8\}$ have $\sim$61%, $\sim$41%, and $\sim$31% of the parameters of a same-width non-looped model.[1] For example, at s=10, $N \in \{98.3, 59.8, 40.2, 30.3\}$ M.

In terms of compute, the looped transformer executes the prelude once, the recurrent block r times (each preceded by the injection layer), and the coda once, so per-token forward FLOPs are:

$$F_{\text{fwd}}(r) = 2(N_{\text{once}} + r\, N_{\text{rec}}) \approx F_{\text{fwd}}(1) = 2\,\ell_{\text{eff}}\, n_b = 2N. \tag{1}$$

Thus, all variants use the same F_{fwd} up to a small injection layer overhead of $r/120 \in \{1.7\%, 3.3\%, 6.7\%\}$ at $r \in \{2, 4, 8\}$.[2] Training FLOPs are $F_{\text{train}}(r) = 3\, F_{\text{fwd}}(r)$, so at a fixed compute budget C every variant trains on approximately the same token count $D \approx C/F_{\text{train}}$.

3.2 Joint Scaling Law

The standard Chinchilla scaling law is defined as

$$L(N, D) = E + AN^{-\alpha} + BD^{-\beta}, \tag{2}$$

where L is validation loss (nats), N the non-embedding parameter count, D training tokens, E the irreducible loss, and A, B, α, β fitted constants. The law assumes loss decreases with higher N and D. The two terms split the loss into a parameter contribution $AN^{-\alpha}$ and a token contribution $BD^{-\beta}$, with α, β setting how fast each diminishes and (via their ratio) the compute-optimal allocation between N and D [10]. Our iso-compute design holds D approximately fixed across r but reduces N, so the same token count is spread over fewer parameters as r grows.

To isolate how much the looped parameters N_{rec} effectively contribute per recurrence, we extend the Chinchilla form (Equation 2) with a *recurrence-equivalence exponent* $\varphi \geq 0$ acting on N_{rec}:

$$L(N_{\text{once}}, N_{\text{rec}}, D, r) = E + A\left(N_{\text{once}} + r^{\varphi} N_{\text{rec}}\right)^{-\alpha} + B\, D^{-\beta}. \tag{3}$$

We refer to $N_{\text{eff}} = N_{\text{once}} + r^{\varphi} N_{\text{rec}}$ as the effective parameter count. The factor $r^{\varphi} \geq 1$ amplifies the contribution of the looped parameters to loss reduction since they are used multiple times.

Intuitively, N_{eff} is the parameter count of a non-looped model that would match the looped variant's loss at the same D. So, if looped models outperform non-looped models at same N, we attribute the performance gain to looped parameters contributing more than they would in a single forward pass. Note that the law is fit on iso-compute runs, so C enters only implicitly through the data. The non-looped reference matches loss at the same D, not at the same compute. When $\varphi < 1$, the recurrent block's contribution to capacity grows slower with r than its contribution to compute (Equation 1), so looped variants underperform non-looped models in compute-matched comparisons.

φ has two natural reference points. $\varphi = 0$ means the recurrent block contributes the same to loss whether run once or r times. The extra $r-1$ executions add FLOPs without any capacity gain, and increasing r at fixed compute only shrinks N and thus raises loss. $\varphi = 1$ corresponds to the iso-FLOPs non-looped model, where each recurrence contributes as much as equivalent unique parameter blocks, so all four architectures would perform the same. Values $0 < \varphi < 1$ quantify partial recovery. $\varphi > 1$ would require the looped block to contribute more than unique blocks (eg, r=8 outperforms r=4), which we do not observe.

4 Scaling Law Results

4.1 Experimental Details

We train at six compute budgets and various model sizes s=$d_{\text{model}}/64$ and adjust training tokens D accordingly. Although our iso-depth setup only requires architectures at the same size to have the

[1] $N(r) = (4 + 16/r)\, n_b + n_i = (48 + 192/r + 2)\, d^2$, giving $\{146, 98, 74\}\, d^2$ for $r \in \{2, 4, 8\}$ versus $240\, d^2$ at r=1.
[2] The overhead is $2rn_i/(2\ell_{\text{eff}} n_b) = r \cdot 2d^2/(20 \cdot 12d^2) = r/120$.

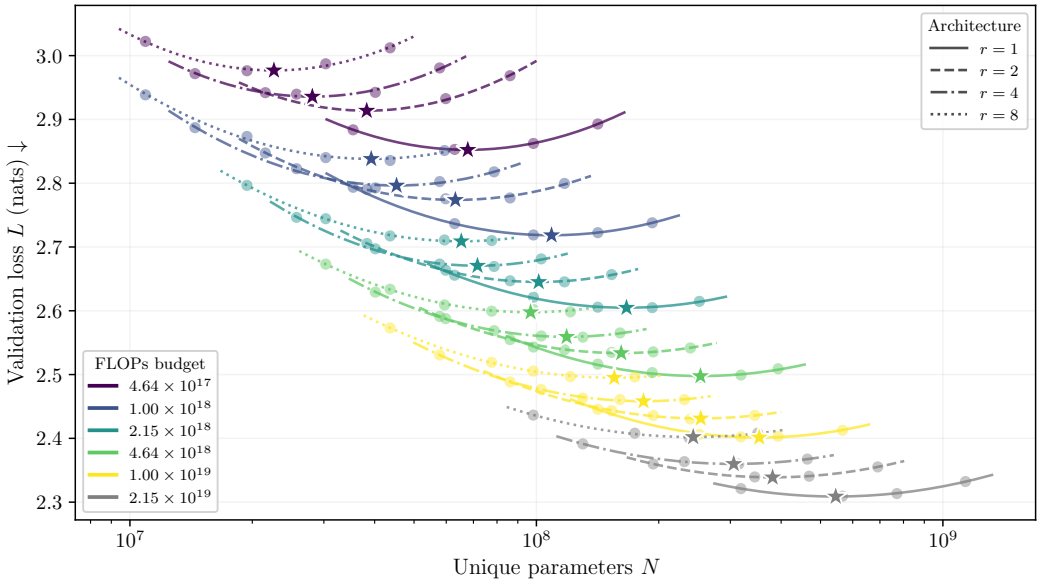


Figure 3: Scaling curves at fixed compute budgets. Curves are per-(budget, r) parabolic fits in $\log N$. Stars mark the fitted compute-optimal (N^*, L^*) points. See Figure 8 for an alternative view.

same effective depth, we hold effective depth fixed at 20 layers throughout this sweep. Appendix G.4 tests 18- and 24-layer configurations. Jointly scaling width and depth introduces additional choices over the layer split between the prelude, recurrent block, and coda, which we discuss in Section 6. At each compute budget, the token count is determined by $D = C/(3F_{\text{fwd}})$, so wider models train on fewer tokens (Table 7). On the resulting 116 pretraining runs, we first fit per-architecture Chinchilla laws, then our proposed joint law with φ.

Implementation. Our implementation builds on nanochat [17]: a decoder-only transformer with RMSNorm [18], RoPE [19], QK normalisation [20], and squared-ReLU MLPs [21]. We use FlashAttention-2 & 3 [22, 23] as the attention backends.

Optimisation. Matrix parameters are optimised with MuonH [24, 25]. Embedding, unembedding, and norm parameters are optimised with AdamW [26]. Weight decay is set to zero (first-order no-op under MuonH's Frobenius-sphere constraint [27]). Learning rates transfer across width, batch size, and training horizon via muP [28] and HyperP [27]. In Appendix E, we sweep base LR and batch size at the reference width and find optima agreeing across architectures (LR regret below 0.005 nats per architecture). All runs use base MuonH LR $\eta_{\text{base}} = 0.014$ and AdamW base LRs 0.3 (embedding), 0.004 (unembedding), and 0.005 (norm), with each LR linearly decayed to 10% of its peak.

Data and validation. Training data is a subset of FineWeb-Edu [29], tokenised with the Llama 2 tokenizer [30] (32,000 base vocabulary) and pre-packed into fixed-length sequences of 2,049 tokens (the extra token provides the next-token target for position 2,048). Thus, all four architectures see exactly the same data stream. Validation loss is reported in nats on a held-out FineWeb-Edu split packed identically to training.

4.2 Per-Architecture Chinchilla Fits

Figure 3 shows validation loss against unique non-embedding parameters N at our grid. At fixed compute C all four architectures trace an approximately parabolic iso-FLOPs curve in $\log N$. The parabolas are systematically offset upward for larger r, with looped minima at wider s than the non-looped baseline (Appendix Figure 8). We fit the standard Chinchilla scaling law (Equation 2) separately for each architecture to characterise individual scaling behaviour and limitations.

Table 1: Chinchilla scaling-law fit parameters per architecture. Amplitudes A, B are rounded to 2 significant figures as they are only loosely identified under iso-Compute [33].

Arch.	A	α	B	β	E	Huber ($\times 10^{-5}$)	R^2
r=1	58	0.251	150	0.267	1.56	5.84	0.9979
r=2	33	0.216	910	0.365	1.60	5.27	0.9983
r=4	23	0.191	1300	0.388	1.56	6.81	0.9976
r=8	41	0.235	780	0.362	1.69	5.33	0.9980

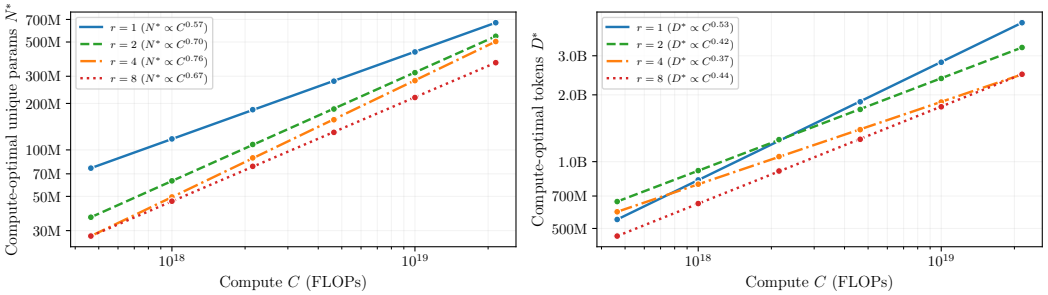


Figure 4: Compute-optimal allocation per architecture. Left: optimal unique parameter count $N^*(C)$ with fitted exponent in the legend. Right: optimal training tokens $D^*(C)$.

Fitting protocol. We follow Hoffmann et al. [10] and minimise the Huber loss [31] ($\delta = 10^{-3}$) between predicted and empirical *log* validation loss using L-BFGS [32]. Specifically, we parameterise the law in log-space ($a = \log A$, $b = \log B$, $e = \log E$) and minimise

$$\mathcal{L}(a, \alpha, b, \beta, e) = \sum_i \text{Huber}_\delta\big(\text{LSE}\big(a - \alpha \log N_i,\ b - \beta \log D_i,\ e\big) - \log L_i\big)\,, \tag{4}$$

where LSE is log-sum-exp. The log-space objective treats relative errors uniformly across the wide dynamic range of N and D. Because the objective is non-convex, we take the best of 500 random L-BFGS-B restarts (details in Appendix G). Table 1 reports the fitted parameters per architecture, all achieving $R^2 > 0.997$ (predicted-vs-actual scatter and residuals in Appendix G).

Compute-optimal allocation and gap. The fitted exponents in Table 1 characterise each architecture's scaling behaviour in isolation but are not directly comparable across r. The unique parameter count N has different semantics under parameter sharing (each parameter in the recurrent block contributes r times for looped models), and iso-FLOPs sampling places each architecture in a different region of the (N, D) plane, where α and β are weakly identified [33]. The compute-optimal loss frontier $L_r^*(C)$, by contrast, is directly comparable across r. We derive the optimal parameter and token allocation $N^*(C), D^*(C)$ by minimising $L(N, C/F(N))$ over N at each budget, with $F(N)$ the architecture's empirical per-token FLOPs at N. The results are shown in Figure 4. Looped optima favour wider models than the non-looped baseline, but their unique parameter count is still lower (left). The optimum compensates for parameter sharing by widening, which raises per-token FLOPs and therefore lowers the optimal training-token count (right).

The resulting loss frontier trails the baseline by $[0.03, 0.06]$ nats at r=2, $[0.05, 0.08]$ nats at r=4, and $[0.09, 0.12]$ nats at r=8, growing monotonically with r across the six budgets. The gap widens at lower budgets and flattens at our largest ($\Delta \leq 0.006$ nats between 10^{19} and 2.15×10^{19} FLOPs).

Extrapolation beyond the grid. To test whether the gap persists past our grid, we train r=1 and r=4 models at s=34 on 47B tokens ($\sim 4 \times 10^{20}$ FLOPs, $\sim 20\times$ the top of our grid). The looped model trails by 0.061 nats in validation loss, inside the $[0.05, 0.08]$ nats r=4 band measured across the grid. See Appendix I.4 for detailed results.

4.3 Joint Scaling Law Fit φ

We now fit the joint law of Equation 3 across all 116 runs. The recurrence-equivalence exponent φ places every architecture on a common $(N_{\text{once}} + r^\varphi N_{\text{rec}}, D)$ surface and measures how much each

Table 2: Joint $(N_{\text{once}}, N_{\text{rec}}, D, r)$ scaling law (Equation 3) fit. The free-φ row reports 95% block-bootstrap CIs (200 resamples of (budget, architecture) cells) below the φ point estimate. Amplitudes A, B are only loosely identified under iso-compute designs [33] and are omitted from the table.

Form	α	β	E	φ	R^2
Joint, φ free	0.199	0.369	1.57	0.459 [0.41, 0.53]	0.9972
Restricted ($\varphi = 0$, pure sharing cost)	0.227	0.390	1.71	0.00	0.9858
Restricted ($\varphi = 1$, non-looped equivalence)	0.218	0.410	1.66	1.00	0.9552

recurrence amplifies the contribution of N_{rec}. This also reduces the four per-architecture fits above (20 parameters) to one shared law with 6 parameters $(A, \alpha, B, \beta, E, \varphi)$. We minimise the same Huber-on-log objective as Equation 4. The results are shown in Table 2.

We measure $\varphi = 0.46$, well below the non-looped reference of $\varphi = 1$ but clearly above 0. Each recurrence amplifies the looped parameters' contribution to loss reduction by a factor of $r^{0.46}$, so the recurrent block contributes $\sim$1.9$\times$ its unique parameter count at r=4 and $\sim$2.6$\times$ at r=8. This matches the qualitative pattern from the compute-optimal allocation above. Each recurrence increases the capacity of the looped parameters ($\varphi > 0$), requiring fewer tokens to reach equal performance at same N. That is why the optimum can spend a smaller share of its FLOPs on tokens and more on widening the model. However, the added capacity is only partial ($\varphi < 1$): the $r^{0.46}$ amplification of the looped block's contribution does not keep up with the linear r increase in its FLOPs cost. Per-FLOP capacity therefore falls with r, and the looped frontier trails the non-looped baseline.

Robustness. The 95% block-bootstrap CI on φ is $[0.41, 0.53]$, with no resample reaching $\varphi = 0$ or $\varphi = 1$. The R^2 values in Table 2 look uniformly high because most variance across the runs comes from the compute-budget axis (cross-architecture loss spans only $\sim$0.1 nats). On that scale, small drops from 0.997 (free φ) are substantial. Figure 1 (right) visualises this fit-quality gap for $\varphi = 1$. Per-r residuals (Appendix Table 8) are comparable across architectures (RMSE 0.009–0.011 nats). Refits on the low-budget half ($C \leq 2.15 \times 10^{18}$) and high-budget half ($C \geq 4.64 \times 10^{18}$) give $\varphi = 0.44$ and $\varphi = 0.49$, both inside the CI. See the full analysis in Appendix G.

5 Case Studies

The fitted φ=0.46 describes our baseline recipe and architecture. We also measure φ under two candidate interventions: truncated backpropagation (a training-recipe change) and hyperconnections (an architectural change). For each case study we rerun the iso-FLOPs grid for $r \in \{2, 4, 8\}$ at our four lower budgets (4.64×10^{17} to 4.64×10^{18} FLOPs), reuse the unchanged r=1 runs as the baseline, and refit the joint law (Table 3). The resulting $\Delta\varphi$ is a single-number summary of how much each recurrence gains or loses capacity under the method. Implementation details are in Appendix H.

5.1 Truncated Backpropagation

Truncated backpropagation through time (BPTT) is a training-efficiency method that is commonly applied to looped transformers [3, 7, 9]. Under full BPTT, gradients flow backward through all r recurrences. Truncated BPTT detaches the recurrent state for all but the last r_{bwd} loops, so the early recurrences run forward only and skip the backward pass. We follow Prairie et al. [9] and set $r_{\text{bwd}} = \lceil r/2 \rceil$. In our setup, the skipped backward passes save roughly 30% of the per-token training FLOPs, allowing more training tokens at fixed budget.

The scaling curves in Figure 5 (left) show that truncated BPTT substantially lowers validation loss across all runs, suggesting at first glance that the method works well. However, the measured φ disagrees, dropping from 0.45 to 0.38 (Table 3). So each additional recurrence is now worth much less in unique-parameter terms than before (smaller r^{φ} in N_{eff}, so matching the same N_{eff} now requires a larger N_{rec}). This likely reflects early loops no longer receiving an accurate learning signal. Evidence for this is the r=2 architecture, which consistently has the largest residual error (Figure 11). Here r_{bwd}=1, so the shared block receives a direct gradient only from the second recurrence. The first recurrence still runs forward and conditions the second loop's input, but this indirect signal is

Table 3: Joint-law refits for each case study alongside the full-BPTT baseline. The baseline was refitted to the four lower budgets. The truncated-BPTT (excl. $r{=}2$) row excludes the $r{=}2$ model, which is an outlier because its shared block receives a direct gradient from only one recurrence.

Variant	α	β	E	φ	R^2
Baseline (full BPTT, linear injection)	0.266	0.457	1.85	0.453	0.9959
Truncated BPTT	0.255	0.484	1.87	0.380	0.9827
Truncated BPTT (excl. $r{=}2$)	0.265	0.492	1.89	0.373	0.9958
Hyperconnections	0.308	0.464	1.93	0.646	0.9900

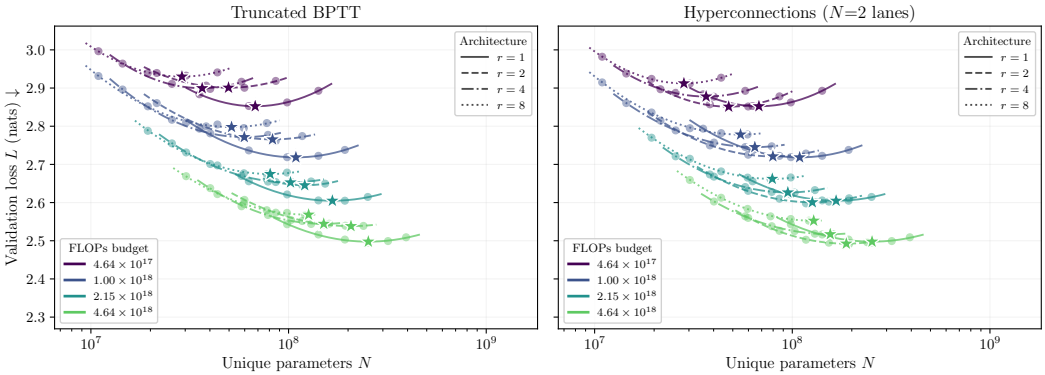


Figure 5: Scaling curves under the two case studies. Thin curves are per-(budget, r) parabolic fits in $\log N$. Stars mark the fitted compute-optimal (N^*, L^*) points.

evidently too weak to train the looping mechanism. The same applies to the forward-only loops at $r{=}4$ and $r{=}8$, which lowers the per-recurrence utility and pulls φ down. Refitting without $r{=}2$ leaves φ essentially unchanged at 0.37 (Table 3), confirming that the capacity drop is not an $r{=}2$ artifact. Overall, the joint law attributes the validation-loss improvement to a larger token budget and a wider compute-optimal model that offsets the per-loop capacity loss but raises inference cost.

5.2 Hyperconnections

We replace the linear input injection of our baseline model with hyperconnections [11], which scale and mix K parallel residual lanes across loops. We hypothesize that better information flow inside the recurrent block leads to a better loop mechanism. We use $K{=}2$ lanes and full BPTT.

The scaling curves in Figure 5 (right) show that hyperconnections substantially lower validation loss across all looped runs, and φ jumps from 0.45 to 0.65 (Table 3). Each additional recurrence is now worth much more in unique-parameter terms than before. Interestingly, the $r{=}2$ architecture even matches or beats the $r{=}1$ baseline at some budgets. Note that $\varphi{=}1$ would require all four architectures to lie on the same compute-optimal frontier, not just $r{=}2$ beating $r{=}1$ at a few budgets.

Thus, hyperconnections are a genuine architectural improvement, as validation loss falls and the compute-optimal allocation moves to narrower widths, lowering per-token inference FLOPs (the opposite of the widening seen under truncated BPTT). However, hyperconnections could in principle be applied to the $r{=}1$ baseline as well, replacing the residual connections between transformer blocks [11]. Our case study applies them only at the loop boundary, which is the natural site for a within-loop intervention. An $r{=}1$ baseline modified the same way would likely shift the calibration of φ. However, our primary comparisons are between looped models, with the non-looped baseline serving only as a calibration anchor. The shift would also likely be modest: concurrent work [16] shows loop-level hyperconnections beat layer-level hyperconnections.

6 Discussion

Per-recurrence value. Our φ=0.46 quantifies how much one recurrence is worth in equivalent unique parameters. At r=4 the shared block recovers $4^{0.46}$≈1.89 unique blocks of capacity, about 47% of full equivalence (φ=1). We asked whether a block looped r times can replace r non-looped blocks at matched compute. Our measurement says no, not under our recipe. But φ is not a property of looping in general. It reflects the joint state of architecture, optimiser, and training recipe, and the case studies of Section 5 show φ can move substantially under either type of change.

$\Delta\varphi$ as a development metric. $\Delta\varphi$ separates token-side from architecture-side gains. Our two case studies illustrate the two cases. Truncated BPTT lowers loss on the token channel but drops φ, while hyperconnections lower loss on the capacity channel by raising φ. We therefore recommend $\Delta\varphi$ as a diagnostic tool alongside validation loss for any new looped LM recipe or architecture. Measuring it is relatively cheap. Per architecture, a focused case study of ∼20 runs across our four lower budgets totals $\sim 5\times 10^{19}$ FLOPs. The same runs also yield the per-architecture compute-optimal allocation. Other methods worth quantifying include shrinking the shared fraction (larger prelude/coda), per-token adaptive compute [5, 6], retrofitting pretrained non-looped models [7, 8], and training with a diffusion objective in place of unrolling the loops [34].

Inference cost. φ also predicts inference cost. A higher φ means each loop adds more capacity, so the same loss can be reached with fewer unique parameters. The freed compute can go into more training tokens, giving a narrower compute-optimum and lower per-token inference FLOPs. A lower φ has the opposite effect. Each loop adds less, so the optimum can widen to compensate and trains on fewer tokens, raising inference cost. A method that substantially lowers φ should therefore be treated cautiously, since the same validation loss is reachable more cheaply with fewer, more powerful recurrences. This is exactly what we observe for truncated BPTT (Section 5.1).

Scaling width and depth. Our iso-depth setup fixes the effective depth across recurrence counts at a given model size, so the r=$\{1, 2, 4, 8\}$ architectures have approximately the same per-token compute (Appendix D). It does not constrain the depth used at different model sizes, so an iso-depth sweep can scale width and depth together as long as the depth remains matched across r at each size. Scaling the number of layers in a looped model is complicated by how those layers are split between the prelude, recurrent block, and coda. A layer inside the recurrent block is executed r times, while prelude and coda layers are executed once, so layer count and effective depth scale differently depending on where a layer is added. In our primary sweep, we vary width and hold effective depth fixed at 20 layers to isolate parameter sharing from these architectural choices. Appendix G.4 tests model scaling with 18 and 24 effective layers and yields a similar φ=0.47, supporting our results.

Downstream validation. Applying our scaling law to downstream continuation loss instead of validation loss yields a task-specific φ_{axis}, which quantifies how much one recurrence is worth on a particular task type. We evaluate this task-specific formulation on the four downstream axes described in Appendix I. For parametric knowledge and reading comprehension, φ_{axis} remains below 1 but is higher than the validation-loss exponent, indicating a smaller parameter-sharing cost (Table 10). The downstream case-study fits also confirm the validation-loss findings and preserve the same relative ordering: truncated BPTT lowers φ_{axis}, while hyperconnections raise it (Table 11). Math word problems and reasoning primitives show no consistent differences beyond run-to-run variation at our compute budgets, including the larger extrapolation runs. The relationship between φ and reasoning performance therefore remains unresolved. A reasoning-heavy pretraining mix might make reasoning differences between the architectures measurable at our budgets [14].

Limitations. Our iso-depth setup limits recurrences at r_{max}=16. The r^{φ} form is a pre-saturation local approximation valid in this range and does not include the architectural ceiling. The joint law also assumes that each additional recurrence must either consistently help ($\varphi > 1$) or consistently hurt ($\varphi < 1$) compared to the non-looped baseline. When some r outperform the baseline and others fall below it, no single φ captures both directions. The non-looped baseline is therefore an important calibration anchor for φ. φ obtained with different baselines are not comparable. Thus, we tune all four architectures with the same recipe in Section 4.1, with Appendix E confirming LR optima agree.

7 Conclusion

We measure the parameter-sharing cost of looped language models as a single number, the recurrence-equivalence exponent φ. On our baseline recipe we obtain φ=0.46, so each shared recurrence is worth roughly half a unique block. The fitted φ responds to design choices, and our two case studies move it in opposite directions even though both lower validation loss. Hyperconnections raise φ to 0.65, while truncated BPTT lowers it to 0.38 by re-allocating compute toward more tokens and a wider model. Raw validation loss does not separate these two cases, but φ does. We therefore propose $\Delta\varphi$ as a metric alongside validation loss for any new looped LM recipe or architecture.

Acknowledgments and Disclosure of Funding

KS and GK received support from the Bavarian Collaborative Research Project PRIPREKI of the Free State of Bavaria Funding Programme "Artificial Intelligence – Data Science".

KS received support from the German Ministry of Education and Research (BMBF) under grant number 01ZZ2316C (PrivateAIM).

The authors gratefully acknowledge the scientific support and resources of the AI service infrastructure LRZ AI Systems provided by the Leibniz Supercomputing Centre (LRZ) of the Bavarian Academy of Sciences and Humanities (BAdW), funded by the Bayerisches Staatsministerium für Wissenschaft und Kunst (StMWK), primarily through the BayernKI initiative.

We acknowledge EuroHPC Joint Undertaking for awarding us access to MareNostrum5 at BSC, Spain.

The authors declare no competing interests.

References

[1] 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.

[2] Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 11398–11442. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

[3] Jonas Geiping, Sean Michael 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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=S3GhJooWIC.

[4] Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling Latent Reasoning via Looped Language Models, November 2025.

[5] Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=QuqsEIVWIG.

[6] Tianyu Fu, Yichen You, Zekai Chen, Guohao Dai, Huazhong Yang, and Yu Wang. Think-at-hard: Selective latent iterations to improve reasoning language models, 2025. URL https://arxiv.org/abs/2511.08577.

[7] 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, 2025. URL https://arxiv.org/abs/2511.07384.

[8] Yeskendir Koishekenov, Aldo Lipani, and Nicola Cancedda. Encode, think, decode: Scaling test-time reasoning with recursive latent thoughts, 2025. URL https://arxiv.org/abs/2510.07358.

[9] 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.

[10] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

[11] Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9FqARW7dwB.

[12] Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=din0lGfZFd.

[13] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

[14] Jonas Knupp, Jan Hendrik Metzen, Jeremias Bohn, Georg Groh, and Kristian Kersting. Depth-recurrent attention mixtures: Giving latent reasoning the attention it deserves, 2026. URL https://arxiv.org/abs/2601.21582.

[15] Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RzYXb5YWBs.

[16] Abbas Zeitoun, Lucas Torroba-Hennigen, and Yoon Kim. Hyperloop transformers, 2026. URL https://arxiv.org/abs/2604.21254.

[17] Andrej Karpathy. nanochat: The best ChatGPT that $100 can buy, 2025. URL https://github.com/karpathy/nanochat.

[18] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

[19] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

[20] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme Ruiz, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd Van Steenkiste, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Collier, Alexey A. Gritsenko, Vighnesh Birodkar, Cristina Nader

Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetic, Dustin Tran, Thomas Kipf, Mario Lucic, Xiaohua Zhai, Daniel Keysers, Jeremiah J. Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7480–7512. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/dehghani23a.html.

[21] David R. So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V. Le. Primer: searching for efficient transformers for language modeling. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

[22] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

[23] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: fast and accurate attention with asynchrony and low-precision. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

[24] Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them ii: Hyperball optimization, 2026. URL https://arxiv.org/abs/2606.16899.

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

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[27] Liliang Ren, Yang Liu, Yelong Shen, and Weizhu Chen. Rethinking language model scaling under transferable hypersphere optimization, 2026. URL https://arxiv.org/abs/2603.28743.

[28] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17084–17097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8df7c2e3c3c3be098ef7b382bd2c37ba-Paper.pdf.

[29] Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

[30] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

[31] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964. doi: 10.1214/aoms/1177703732. URL https://doi.org/10.1214/aoms/1177703732.

[32] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980. ISSN 00255718, 10886842. URL http://www.jstor.org/stable/2006193.

[33] Tamay Besiroglu, Ege Erdil, Matthew Barnett, and Josh You. Chinchilla scaling: A replication attempt, 2024. URL https://arxiv.org/abs/2404.10102.

[34] Makoto Shing, Masanori Koyama, and Takuya Akiba. Diffusionblocks: Block-wise neural network training via diffusion interpretation, 2026. URL https://arxiv.org/abs/2506.14202.

[35] Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond chinchilla-optimal: accounting for inference in language model scaling laws. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[36] Nicholas Roberts, Sungjun Cho, Zhiqi Gao, Tzu-Heng Huang, Albert Wu, Gabriel Orlanski, Avi Trost, Kelly Buchanan, Aws Albarghouthi, and Frederic Sala. Test-time scaling makes overtraining compute-optimal, 2026. URL https://arxiv.org/abs/2604.01411.

[37] Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise loRA. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=WwpYSOkkCt.

[38] Kristian Schwethelm, Daniel Rueckert, and Georgios Kaissis. Depth-adaptive inference of looped language models via continuous depth batching, 2026. URL https://arxiv.org/abs/2608.09444.

[39] Gemma Team. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015. doi: 10.1109/ICCV.2015.123.

[41] Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Kumar Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee F Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Kamal Mohamed Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Joshua P Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah M Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham M. Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander T Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alex Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-LM: In search of the next generation of training sets for language models. In *The Thirty-eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=CNWdWn47IE.

[42] David Heineman, Valentin Hofmann, Ian Magnusson, Yuling Gu, Noah A. Smith, Hannaneh Hajishirzi, Kyle Lo, and Jesse Dodge. Signal and noise: A framework for reducing uncertainty in language model evaluation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=sAFottNlra.

[43] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada,

July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

[44] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

[45] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1160/.

[46] Aarohi Srivastava, Abhinav Rastogi, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj.

[47] 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.

[48] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

[49] Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020. doi: 10.1162/tacl_a_00317. URL https://aclanthology.org/2020.tacl-1.30/.

[50] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124/.

[51] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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)*, pages 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246/.

[52] Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019. doi: 10.1162/tacl_a_00266. URL https://aclanthology.org/Q19-1016/.

[53] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/.

[54] Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing English math word problem solvers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.92. URL https://aclanthology.org/2020.acl-main.92/.

[55] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1136. URL https://aclanthology.org/N16-1136/.

[56] Nikunj Saunshi, Stefani Karp, Shankar Krishnan, Sobhan Miryoosefi, Sashank J. Reddi, and Sanjiv Kumar. On the inductive bias of stacking towards improving reasoning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=3ZAfFoAcUI.

A Extended Related Work

Scaling laws. Kaplan et al. [13] established power-law relations between loss, model size, and training tokens. Hoffmann et al. [10] refined the allocation (Chinchilla) and found that compute-optimal training scales parameters and tokens at roughly equal rates with compute. Subsequent work has examined learning-rate transfer [27, 28] and inference-aware scaling that trades training tokens for inference cost [35, 36]. We extend these analyses to looped architectures.

Iso-parameter scaling law. Concurrent work by Prairie et al. [9] fits an iso-parameter scaling law at fixed unique parameter count N. Iso-parameter and iso-depth formulations impose different inference constraints. For example, at width $s{=}34$ and 4k context length, iso-depth scaling reduces weight memory from 2.55GB to 0.98GB at $r{=}8$, while per-token FLOPs vary by less than 5% and KV-cache memory remains constant. The corresponding iso-parameter model retains approximately same weight memory across all architectures but executes $6.6\times$ as many layers, uses $6.4\times$ the per-token FLOPs, and requires $6.6\times$ the KV-cache memory. Appendix D reports the full comparison.

Additionally, three methodological details affect direct comparison. (1) Prairie et al. [9] sample per-step recurrence counts from a distribution with mean μ_{rec}, while we fix r architecturally. (2) They use truncated BPTT, which reduces training FLOPs. We train our main grid under full BPTT to keep training and inference FLOPs aligned with the matched non-looped baseline, and probe the truncated-BPTT alternative separately (Section 5.1 shows it lowers φ). (3) Their default input injection is diagonal ($\mathcal{O}(d)$ parameters). Ours is a linear map $W_{\text{inject}} \in \mathbb{R}^{d\times 2d}$ with a small FLOPs cost. Their diagonal-injection layer remains untested in our framework. Overall, the two scaling laws are complementary and answer different questions.

A scaling law that matches parameters, FLOPs, and memory simultaneously remains an open direction. Knupp et al. [14] achieve such matching, introducing FLOP-neutral capacity through sparse MoEs alongside depth attention and per-recurrence expert routing layered onto every block. Extending their method to a scaling law setup would be interesting future work.

Depth-adaptive inference. A main promise of looped LMs is depth-adaptive inference. Tokens in a sequence can exit after different numbers of recurrences, spending less compute on easy predictions and more on difficult ones [1, 6]. Realising this requires additional choices during both training and inference. Learned exit gates add training FLOPs [4], while recurrence-count sampling makes training compute variable and complicates FLOP accounting [3]. At inference, variable exit depths break uniform batching and require specialised scheduling and KV-cache handling [5, 37]. Continuous depth batching provides these mechanisms and enables wall-clock speedups [38]. Predetermined routing schemes such as Mixture-of-Recursions [5] and LoopFormer [15] provide an alternative with more uniform execution. A scaling law that jointly accounts for recurrence value, gate training, variable inference compute, and serving efficiency is an important direction for future work.

Test-time loop extrapolation. The optimistic picture from synthetic algorithmic tasks [2], of training at low recurrence counts and extrapolating to greater depth at inference, has not transferred cleanly to general language modelling. Geiping et al. [3], Prairie et al. [9] train their models with sampled recurrence counts extending to large values, to enable test-time scaling. However, Prairie et al. [9] fit a joint training–inference scaling law whose test-time component is a saturating exponential $\mathcal{L}(T) = \mathcal{L}_\infty + Z\exp(-zT/\mu_{\text{rec}})$ that plateaus at $T \approx \mu_{\text{rec}}$. Thus, the mean recurrence in training caps the test-time frontier. Ouro [4] was trained with an anytime-prediction gate and reports no inference-time gains beyond the trained depth either. Taken together, effective inference depth in trained looped LMs concentrates near the training depth distribution rather than extrapolating freely past it. We therefore treat r as an architectural, not a test-time, scaling axis. Until depth extrapolation works, it does not raise the worth of a recurrence beyond what φ captures at training time.

B Compute Resources

Experiments ran on a mix of A100-80GB and H100-80GB GPUs. The 116-run main grid, two case studies, and the effective-depth study account for the bulk of the budget. The $s{=}34$ extrapolation pair at 47B tokens also added substantial cost. The full project, including exploratory configurations, failed runs, and side experiments on the same accounts, consumed approximately $5{,}000$ GPU-hours.

Table 4: Transformer architecture.

Component	Details
Sequence length	$T = 2048$ tokens
Attention	Full causal self-attention, no sliding window, no dropout
Attention backend	FlashAttention-2 [22] on A100, FlashAttention-3 [23] on H100
Position encoding	Rotary embeddings [19], base $\theta = 10{,}000$
Head dimension	$d_{\text{head}} = 128$, $n_{\text{head}} = d_{\text{model}}/128$
QK normalisation	Functional RMSNorm on q, k before attention [20]
MLP activation	Squared ReLU [21], hidden dim $= 4d_{\text{model}}$
Normalisation	Learnable RMSNorm [18], pre-norm, ϵ=10^{-6}
Biases	None (all linear layers bias-free)
Embeddings	Untied `wte` and `lm_head`, token embeddings cast to bf16
Vocabulary	Llama 2 tokenizer [30], padded $32008 \to 32064$
Logit softcap	$z = 15 \cdot \tanh(\text{logits}/15)$ [39], applied in fp32 before the loss
Dropout	None

C Model Architecture

C.1 Implementation Details

All architectures are decoder-only transformers built on top of nanochat [17], using the same pre-norm block summarised in Table 4. The layer partition for each r is $\ell_{\text{prelude}} + r \cdot \ell_{\text{recur}} + \ell_{\text{coda}} = 20$ with $(\ell_{\text{prelude}}, \ell_{\text{coda}}) = (2, 2)$ for $r > 1$, so $\ell_{\text{recur}} = 16/r$ evaluates to $\{8, 4, 2\}$ for $r \in \{2, 4, 8\}$. Figure 2 right visualises the four architectures.

Each transformer block computes

$$\hat{x} = x + \text{Attn}(\text{RMSNorm}(x)), \qquad x_{\text{out}} = \hat{x} + \text{MLP}(\text{RMSNorm}(\hat{x})).$$

In addition to the pre-norms inside each block, three model-level RMSNorms are applied on the residual stream: one after the token embedding, one at the end of every recurrence iteration (so the state handed to the next iteration or to the coda has controlled scale), and one before the `lm_head`.

Input injection (looped only). For each looped architecture ($r > 1$), every recurrence iteration begins with a linear injection step

$$u^{(t)} = W_{\text{inject}}\,[e \,\|\, h^{(t)}], \qquad W_{\text{inject}} \in \mathbb{R}^{d \times 2d}, \tag{5}$$

where e is the prelude output (constant across recurrences), $h^{(t)}$ is the recurrent state at iteration t with $h^{(0)} = e$, and W_{inject} is initialised as $[I \,\|\, 0]$ so that $u^{(0)} \approx e$ at the start of training. Appendix C.2 ablates this choice against additive-residual and no-injection variants.

Initialisation. Token embeddings are drawn from $\mathcal{N}(0, 1)$ and then cast to bf16. The LM head is $\mathcal{N}(0, 10^{-3})$. Attention and MLP weights use $\mathcal{U}(-a, a)$ with $a = \sqrt{3}/\sqrt{d_{\text{model}}}$ (equivalently, the same standard deviation $1/\sqrt{d_{\text{model}}}$ as the matched normal but with bounded tails), except `mlp.c_proj` which uses $a = \sqrt{3}/\sqrt{4d_{\text{model}}}$ (Kaiming fan-in [40] over its input width $4d_{\text{model}}$). The injection layer is initialised as $[I \,\|\, 0]$, and all RMSNorm scales are initialised to one.

Relation to Huginn. We adopt the prelude-recur-coda architecture and learned concatenation-based input injection of Huginn [3]. Huginn additionally samples recurrence counts during training, initialises the recurrent state randomly from $\mathcal{N}(0, 1)$, and uses truncated BPTT. The first two choices primarily support adaptive inference: recurrence-count sampling trains the model to predict after different numbers of loops, while random state initialisation stabilises the loop when increasing the recurrence count at inference. Reliable improvements from increasing recurrence at inference remain an open question, which is outside the scope of this work (Appendix A). Our goal is to isolate the parameter-sharing cost at a fixed recurrence count. We therefore avoid the added stochasticity of random state initialisation and the variable per-token training FLOPs introduced by recurrence-count sampling. We use full BPTT so that training FLOPs account for every recurrence, and study truncated BPTT separately in Section 5.1. We ablate random state initialisation below.

Table 5: Input-injection ablations at s=10, r=4, and C=10^{18} FLOPs. Tokens differ across variants because the parameter-free and hyperconnection methods save the linear injection's FLOPs overhead.

Variant	Tokens trained	Val. loss (nats)
Linear (default)	955M	2.793
Linear + random state	955M	2.804
Additive	973M	2.797
Passthrough	973M	7.400
Hyperconnections (K=2)	973M	2.757

C.2 Input-Injection and State-Initialisation Ablations

Our default injection is the linear map of Equation 5, following the concatenation-based input injection [3]. The linear map adds $2d^2_{\text{model}}$ parameters and, applied once per recurrence, a non-negligible FLOPs overhead that is paid back only if it improves quality. We verify this at the reference configuration (s=10, r=4, target compute 10^{18} FLOPs) against two parameter-free alternatives:

- **Passthrough** ($u^{(t)} = h^{(t)}$): no injection, recurrence is depth-only with $h^{(0)}$ initialised from the prelude output.
- **Additive** ($u^{(t)} = h^{(t)} + e$): parameter-free residual injection with $h^{(0)} = 0$, so the first iteration sees $u^{(0)} = e$.

We additionally replace the default initialisation $h^{(0)}$=e with a random initialisation $h^{(0)}\sim\mathcal{N}(0, 1)$ [3], retaining the learned linear injection. All variants use the same target FLOPs budget, so the parameter-free alternatives train on more tokens than the learned-injection variants (973M vs. 955M, a $\sim$2% data advantage from the saved injection FLOPs). Results are summarised in Table 5. Passthrough fails to train, showing that some form of injection is essential at this scale. Additive is competitive but trails the linear injection by 0.004 nats despite its token advantage. Random state initialisation has little effect, trailing the deterministic default by 0.011 nats. We therefore adopt the linear injection with deterministic state initialisation for all reported scaling-law runs. Its FLOPs overhead is accounted for in n_i in Equation 1 and thus included in every iso-FLOPs comparison.

Hyperconnections [11] are only listed for reference. We did not adopt them in the main scaling-law grid. They are used in Section 5.2 to test whether they improve φ.

D Inference Cost under Iso-Depth and Iso-Parameter Scaling

We compare the inference cost of iso-depth and iso-parameter loop scaling at width s=34. Iso-depth reduces the number of layers in the recurrent block as r increases, keeping the effective depth fixed (Figure 2). Iso-parameter scaling retains the full recurrent block, keeping the unique parameter count approximately fixed while effective depth grows with r. Table 6 reports the resulting inference cost of both variants based on weight memory, per-token FLOPs, KV-cache memory, and the maximum number of concurrent sequences that fit on an 80GB accelerator. Under iso-depth scaling, weight memory decreases from 2.55GB to 0.98GB, while per-token FLOPs vary by less than 5% and the KV-cache footprint remains constant. Under iso-parameter scaling, the r=8 model executes 6.6× as many layers as the baseline, uses 6.4× the FLOPs per generated token, and supports 16 instead of 105 concurrent sequences at a 4k-token context length. Iso-depth therefore isolates the parameter-sharing trade-off while controlling the inference costs that grow with recurrence. Iso-depth comparisons can also be constructed at greater effective depths. For example, the 132 effective layers of the iso-parameter r=8 model could be matched by a 132-layer non-looped baseline.

E Hyperparameter Tuning

E.1 Learning Rate Sweep

We sweep the MuonH [24, 27] matrix learning rate at s=10 with a tokens-per-parameter ratio of 10 ($\sim$1B training tokens) and batch size B = 262,144 (256K), independently for each architecture,

Table 6: Inference costs at width s=34 (d_{model}=2176) in bf16. N excludes embedding parameters, weight memory includes them. FLOPs are calculated per generated token at the stated context length, including attention and the output projection. The sequence counts assume an 80GB accelerator with 2GB reserved for activations and other workspace. The looped iso-parameter models match the baseline parameter count up to the additional 10M parameters of the input-injection layer.

					4k context			16k context		
Model	Design	ℓ_{eff}	N (M)	Weights (GB)	GFLOPs/tok.	KV/seq. (GB)	Seqs.	GFLOPs/tok.	KV/seq. (GB)	Seqs.
r=1	Baseline	20	1136	2.55	3.13	0.71	105	5.26	2.85	26
r=2	Iso-depth	20	691	1.66	3.16	0.71	107	5.30	2.85	26
r=4	Iso-depth	20	464	1.21	3.20	0.71	107	5.34	2.85	26
r=8	Iso-depth	20	350	0.98	3.28	0.71	108	5.42	2.85	27
r=2	Iso-param.	36	1146	2.57	5.55	1.28	58	9.40	5.13	14
r=4	Iso-param.	68	1146	2.57	10.37	2.42	31	17.64	9.70	7
r=8	Iso-param.	132	1146	2.57	20.00	4.71	16	34.12	18.82	4

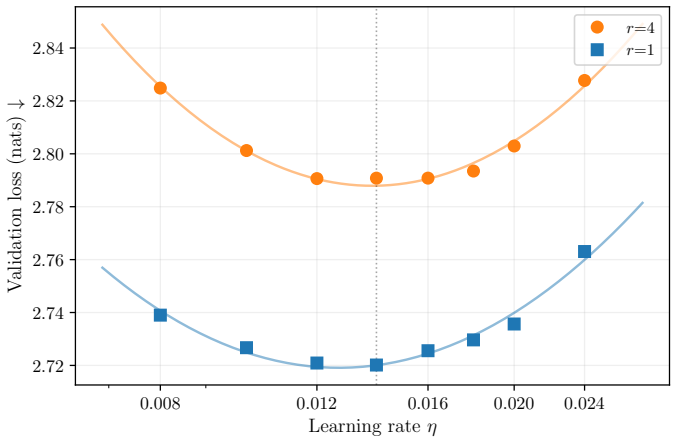


Figure 6: Learning rate sweep at s=10 (ratio 10, $B = 256$K). Both architectures exhibit a clear U-shaped loss landscape with a shared optimum near $\eta^* \approx 0.014$. The dotted vertical line marks $\eta = 0.014$, the base LR adopted for all scaling-law runs.

using eight LR values per architecture in the range $\eta \in [0.008, 0.024]$ with extra density around the optimum. The batch size was chosen from a separate sweep at the same reference configuration over $B \in \{256\text{K}, 512\text{K}, 1\text{M}\}$ tokens, where 256K yielded uniformly lower loss for both architectures.

Both architectures converge to a shared optimum near $\eta^* \approx 0.014$, which we adopt as the base learning rate for all subsequent experiments (Figure 6).

E.2 Transfer Validation

Under the HyperP framework [27] the *base* learning rate η_{base} (the value fed into HyperP before its width and data corrections are applied) should be invariant to both width and training horizon (after the $D^{-0.32}$ data-scaling correction of the HyperP LR rule, with D the training-token count). We verify both claims by repeating the LR sweep under varied conditions and measuring the regret: the loss penalty of using $\eta_{\text{base}} = 0.014$ instead of the per-condition optimum.

Width transfer. We sweep at $s \in \{8, 10, 14\}$ (ratio 10, $B = 256$K). Figure 7 (left column) shows the regret U-curves in base LR space: all minima cluster near $\eta_{\text{base}} = 0.014$ with a maximum regret of 0.004 nats (s=8 looped). As a lightweight sanity check past the sweep range, we run the two candidate LRs $\eta_{\text{base}} \in \{0.012, 0.014\}$ at s=18 for the looped architecture and find 0.014 marginally better (2.473 vs. 2.476 nats), confirming that 0.014 remains near-optimal.

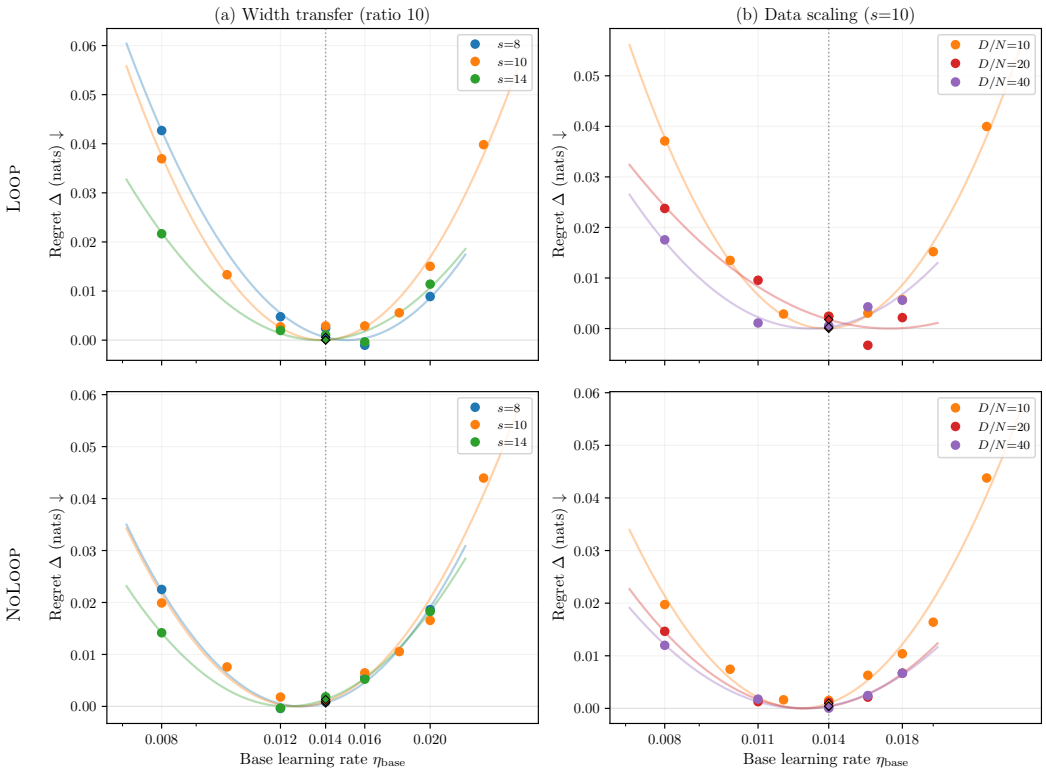


Figure 7: Transfer validation. Regret (loss above the per-condition optimum) vs. base learning rate η_{base}. Vertical dotted line marks $\eta_{\text{base}} = 0.014$; diamond markers show the regret at that point. Rows split by architecture (looped r=4, top; non-looped r=1, bottom). All conditions incur ≤ 0.005 nats regret, so $\eta_{\text{base}} = 0.014$ transfers cleanly across width and training horizon.

Data scaling. We sweep at s=10 ($B = 256$K) with ratios $\{10, 20, 40\}$, spanning a $4\times$ range in training tokens. If the $T^{-0.32}$ exponent is correct, the data-scaling correction adjusts the effective LR automatically and the optimal base LR should remain constant. Figure 7 (right column) confirms this: regret at $\eta_{\text{base}} = 0.014$ stays below 0.005 nats across all ratios for both architectures.

F Model Sizes and Trained Tokens

For each compute budget we sweep model width to find the compute-optimal point at every recurrence count $r \in \{1, 2, 4, 8\}$. Table 7 reports the unique non-embedding parameter count $N(s, r)$ at each width and the training tokens D used at each (budget, width) cell. Looped models train on slightly fewer tokens due to input injection compute overhead. Empty cells are untested widths.

G Scaling Law Fit Diagnostics

We conduct robustness checks for the per-architecture Chinchilla fits (Equation 2) and the joint $(N_{\text{once}}, N_{\text{rec}}, D, r)$ law (Equation 3). Residuals for the per-architecture fits, aggregate residual statistics for the joint fit, the block-bootstrap procedure behind the φ confidence interval, and stability of φ across budget halves.

Optimisation details. The Huber-on-log objective of Equation 4 is non-convex, so for both the per-architecture and joint fits we take the best of 500 random L-BFGS-B restarts. Parameters are constrained to the box $a, b \in [-5, 35]$, $\alpha, \beta \in [0, 2.5]$, $e \in [-3, 2]$ (and $\varphi \in [-3, 3]$ for the joint fit), with starting points drawn uniformly inside the box and a per-restart cap of 10,000 iterations.

Table 7: Model sizes and trained tokens. Left: unique non-embedding parameter count N (M) per (width, recurrence) cell. Right: training tokens (B) per (width, budget) cell for $r = 1$.

s	Unique params N (M)				Training tokens (B) per budget (FLOPs)					
	r=1	r=2	r=4	r=8	$4.64{\cdot}10^{17}$	10^{18}	$2.15{\cdot}10^{18}$	$4.64{\cdot}10^{18}$	10^{19}	$2.15{\cdot}10^{19}$
6	35.4	21.5	14.5	10.9	0.98	2.10	—	—	—	—
8	62.9	38.3	25.7	19.4	0.64	1.36	2.95	—	—	—
10	98.3	59.8	40.2	30.3	0.45	0.97	2.08	4.49	—	—
12	141.6	86.1	57.8	43.7	0.34	0.72	1.55	3.34	7.13	—
14	192.7	117.2	78.7	59.4	—	0.56	1.20	2.59	—	—
16	251.7	153.1	102.8	77.6	—	—	0.96	2.07	4.43	—
18	318.6	193.8	130.1	98.2	—	—	—	1.70	3.62	7.78
20	393.3	239.2	160.6	121.3	—	—	—	1.42	3.05	—
24	566.3	344.5	231.2	174.6	—	—	—	—	2.20	4.71
28	770.8	468.9	314.7	237.7	—	—	—	—	—	3.58
34	1136.5	691.4	464.1	350.4	—	—	—	—	—	2.52

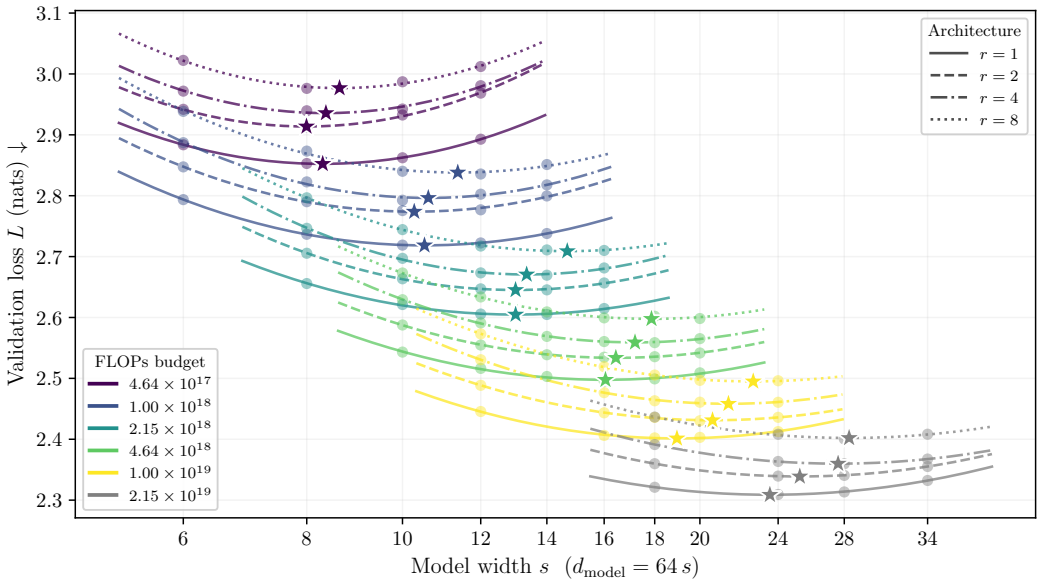


Figure 8: Scaling curves at fixed compute budgets, against model width s. Same runs as Figure 3. Thin curves are per-(budget, r) parabolic fits in $\log s$. Stars mark the fitted compute-optimal (s^*, L^*) points. At matched width, looped models reach higher loss than the non-looped baseline and their optima sit mostly at larger s.

G.1 Per-Architecture and Joint Fit Residuals

Figure 9 shows predicted vs. actual validation loss for the four per-architecture Chinchilla fits. Points cluster tightly around the diagonal with a maximum residual below 0.007 nats. Figure 10 plots the same residuals against N and D and shows no systematic bias across either axis.

On the joint law, which fits all 116 runs with six shared parameters $(A, \alpha, B, \beta, E, \varphi)$ rather than 20 per-architecture parameters, residuals are naturally larger but still small: max $|\text{resid}| = 0.036$ nats, pooled RMSE $= 0.010$ nats, and $R^2 = 0.997$. Table 8 breaks the joint-fit residuals down by architecture. RMSE is comparable across r (0.009–0.011 nats) and per-r means lie within ± 0.006 nats of zero, so a single shared φ fits all four architectures with comparable accuracy and no architecture absorbs a disproportionate share of the residual mass.

Table 8: Joint-law residual statistics broken down by architecture (actual − predicted, in nats). Each architecture contributes 29 runs to the joint fit.

Architecture	n	mean resid	max |resid|	RMSE
r=1	29	−0.001	0.018	0.009
r=2	29	+0.004	0.036	0.011
r=4	29	−0.006	0.023	0.011
r=8	29	+0.001	0.029	0.010

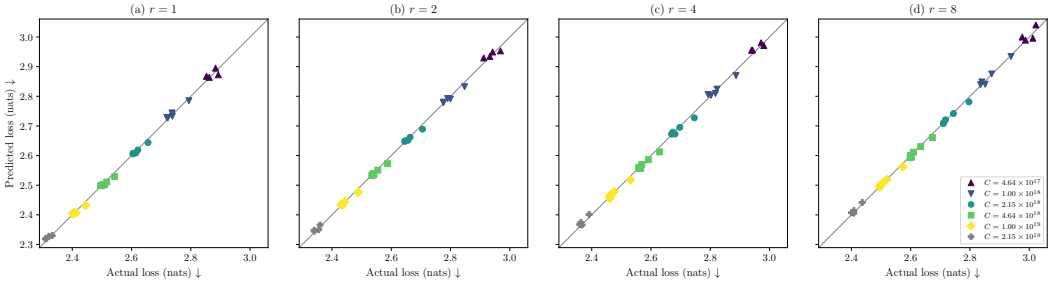


Figure 9: Per-architecture Chinchilla fit quality: predicted vs. actual validation loss, one panel per architecture ($r \in \{1, 2, 4, 8\}$). Markers redundantly encode the compute budget by both shape and colour. Points cluster tightly around the identity line across all four architectures.

G.2 Bootstrap Procedure

The 95% CI reported alongside the joint-law point estimate ($\varphi = 0.46$, $[0.41, 0.53]$) is a block bootstrap over *(budget, architecture) cells*: each cell groups all widths trained at a given (compute budget, r) pair, so resampling respects the experimental block structure rather than treating individual runs as independent. We draw 200 resamples with replacement of the non-empty cells (6 budgets $\times$ 4 architectures), refit the joint law on each resample, and report the 2.5th / 97.5th percentiles of the resulting φ distribution. Zero resamples reach either $\varphi = 0$ or $\varphi = 1$. We do not bootstrap restricted variants of the law.

G.3 Stability Across Budget Halves

Refitting the joint law separately on the low-budget half ($C \leq 2.15 \times 10^{18}$, n=56 runs) gives $\varphi = 0.44$, and refitting on the high-budget half ($C \geq 4.64 \times 10^{18}$, n=60 runs) gives $\varphi = 0.49$. φ therefore does not drift with scale inside our compute window, and the bootstrap CI comfortably contains both half-window estimates.

G.4 Varying Effective Depth

Our primary scaling sweep varies model width while fixing the effective depth at 20 layers. The iso-depth setting only requires architectures compared at the same model width to have equal effective depth, it does not require different model widths to use the same depth. We therefore test whether our main findings depend on the fixed 20-layer design by repeating the $r \in \{1, 2, 4, 8\}$ width sweep at the four smallest compute budgets with effective depths of 18 and 24 layers. Changing the effective depth of a looped model also requires choosing how layers are allocated between the prelude, recurrent block, and coda. We keep the recurrent block's contribution to effective depth fixed at $r\ell_{\text{recur}} = 16$ and set $(\ell_{\text{prelude}}, \ell_{\text{coda}})$ to $(1, 1)$, $(2, 2)$, and $(4, 4)$ for the 18-, 20-, and 24-layer configurations, respectively. Each configuration includes a depth-matched non-looped baseline. The results change little across the three effective depths. Pooling all 240 runs gives a shared recurrence-equivalence exponent of $\varphi = 0.468$ with a bootstrap 95% CI of $[0.442, 0.494]$, close to the primary estimate of 0.46.

G.5 Example: Equivalent Model Sizes at r=4

We compare a looped r=4 model and a non-looped baseline at the same width and effective depth. The resulting unique-parameter and effective-parameter counts are purely architectural ratios.

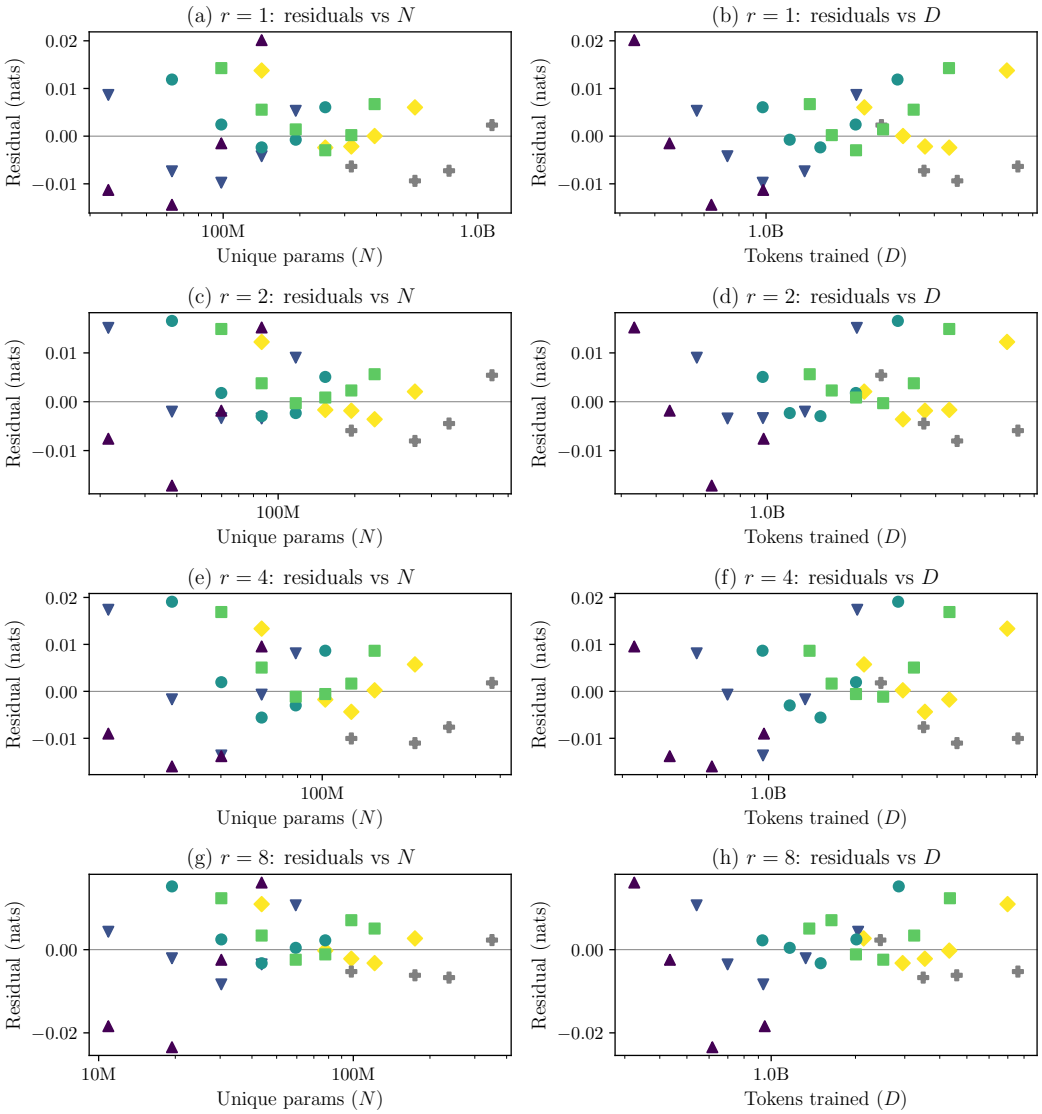


Figure 10: Per-architecture Chinchilla fit residuals (actual − predicted) vs. unique parameters N (left column) and tokens D (right column). Rows are the four architectures $r \in \{1, 2, 4, 8\}$. Markers encode the compute budget by both shape and colour.

Take any $r{=}1$ configuration with $N(r{=}1) = 1$B and width d_{model}. At the same d_{model} and $\ell_{\text{eff}}{=}20$ with $(\ell_{\text{prelude}}, \ell_{\text{coda}}) = (2, 2)$, the $r{=}4$ variant uses $\ell_{\text{recur}}{=}4$ and the unique-parameter ratio (from the $N_{\text{once}}, N_{\text{rec}}$ split in Section 3.1, dropping the small injection term $n_i = n_b/6$) is

$$\frac{N(r{=}4)}{N(r{=}1)} = \frac{(\ell_{\text{prelude}} + \ell_{\text{recur}} + \ell_{\text{coda}})\, n_b}{\ell_{\text{eff}}\, n_b} = \frac{8}{20} = 0.40,$$

so the $r{=}4$ variant has $0.40\times$ the unique parameters of the non-looped model: $\approx$410M including the injection term. The effective-parameter ratio under the joint law (Equation 3) is

$$\frac{N_{\text{eff}}(r{=}4;\varphi)}{N(r{=}1)} = \frac{N_{\text{once}} + 4^{\varphi} N_{\text{rec}}}{\ell_{\text{eff}}\, n_b} \approx \frac{4 + 4^{\varphi} \cdot 4}{20} \overset{\varphi=0.46}{\approx} 0.58,$$

giving the $\approx$580M figure. Per-token training FLOPs depend only on the executed-layer count, which is identical at fixed d_{model} up to the $\sim$3% injection overhead of Equation 1, so the $r{=}4$ model trains at the same per-step cost as the 1B non-looped baseline. The same calculation at any other reference size gives the same percentages.

H Case Study Details

H.1 Methods

Truncated backpropagation. We follow Prairie et al. [9] and set the gradient window to $r_{\text{bwd}} = \lceil r/2 \rceil$ for $r \in \{2, 4, 8\}$, giving $r_{\text{bwd}} \in \{1, 2, 4\}$. The forward pass is unchanged. After the i-th recurrence, the recurrent state $s^{(i)}$ is detached for all $i < r - r_{\text{bwd}}$, so gradients flow only through the last r_{bwd} iterations. Detached iterations skip the backward pass and save roughly half the FLOPs of a full recurrent iteration, which translates to about 30% fewer training FLOPs per token under our prelude-recur-coda partition,

$$F^{\text{trunc}}_{\text{train}}(r) = \big(2(r - r_{\text{bwd}}) + 6r_{\text{bwd}}\big)(\ell_{\text{recur}} n_b + n_i) + 6(\ell_{\text{prelude}} + \ell_{\text{coda}})n_b. \tag{6}$$

The freed compute is reinvested as more tokens at fixed budget. The empirical median ratio across all (budget, r, s) cells is $D_{\text{trunc}}/D_{\text{full}} = 1.315$.

Hyperconnections. We use our own implementation of a looped transformer with hyperconnections [11]. Note that Zeitoun et al. [16] concurrently proposed a similar architecture. We replace the linear input-injection layer with K=2 parallel lane states $\ell^{(i)} \in \mathbb{R}^{K \times d_{\text{model}}}$ mixed at every recurrence iteration. Lanes are initialised by broadcasting the prelude output e across all K slots, $\ell^{(0)} = (e, \ldots, e)$. At iteration $i \in \{0, \ldots, r-1\}$ the recurrent block sees

$$u^{(i)} = \alpha_i \cdot \ell^{(i)}, \qquad s^{(i)} = \text{RecurBlock}(u^{(i)}), \qquad \ell^{(i+1)} = M_i \ell^{(i)} + \beta_i \otimes s^{(i)}, \tag{7}$$

with per-iteration mixing parameters $\alpha_i \in \mathbb{R}^K$, $M_i \in \mathbb{R}^{K \times K}$, $\beta_i \in \mathbb{R}^K$, for a total of $r(K^2 + 2K)$ scalars across all iterations (32 at K=2, r=4). The coda receives the sum-pooled lanes $\sum_k \ell^{(r)}_k$. We adopt the cyclic initialisation of Zhu et al. [11], $\alpha_i = \mathbf{e}_{i \bmod K}$, $M_i = I$, $\beta_i = \mathbf{1}$, so that the first training iteration reduces to a plain looped forward pass. All hyperconnect runs use full BPTT. The per-token training-FLOPs cost of the mixing operations is $6r(K^2 + 2K)d_{\text{model}}$ (forward $2\times$ plus backward $4\times$), accounted for in our budget calculations and negligible: two orders of magnitude smaller than the linear injection it replaces.

H.2 Fit Quality

Figure 11 shows predicted vs. actual validation loss for both case-study joint fits, the analogue of Figure 9 on the main grid. Points cluster on the diagonal in both panels, with overall RMSEs an order of magnitude below the inter-architecture spread.

The truncated-BPTT panel shows where the $R^2 = 0.983$ is lost. Most of the residual mass sits at r=2 ($r_{\text{bwd}} = 1$), where the joint law systematically under-predicts loss. Only a single recurrence receives a direct gradient, which likely undertrains the looping mechanism for that architecture. The $r \in \{4, 8\}$ rows are well-behaved. Refitting on $r \in \{4, 8\}$ alone (Table 3) raises R^2 from 0.983 to 0.996, removing the r=2 residual mass without changing φ.

The hyperconnections panel does not show a comparable architecture-specific structure. Residuals are uniformly small across r, and the fewer runs (83 vs. 116 in the main fit) and narrower compute span (four of six budgets) are the main sources of R^2 loss relative to the main joint fit.

H.3 Compute-Optimal Allocation Under the Case Studies

The shift in φ has predictable consequences for compute-optimal allocation (Figure 5). Treating r as fixed and writing the joint law as $L = E + (A \cdot g_r^{-\alpha})N^{-\alpha} + BD^{-\beta}$ with $g_r = N_{\text{once}}/N + r^{\varphi} N_{\text{rec}}/N$, a higher φ raises g_r at $r > 1$, which lowers the effective parameter amplitude $A_{\text{eff}} = A g_r^{-\alpha}$, which in turn lowers the compute-optimal width $N^*(C)$ for that r.

Truncated BPTT. φ falls from 0.45 to 0.38, so g_r shrinks at every $r > 1$ and the compute-optimal width s^* widens further than under full BPTT. The picture is therefore one of larger looped models, consistent with the rightward shift of the truncated BPTT compute-optimal stars in Figure 5 (left). Wider compute-optimal models also raise per-token inference FLOPs, so the recipe trades training FLOPs for inference FLOPs at fixed deployment budget.

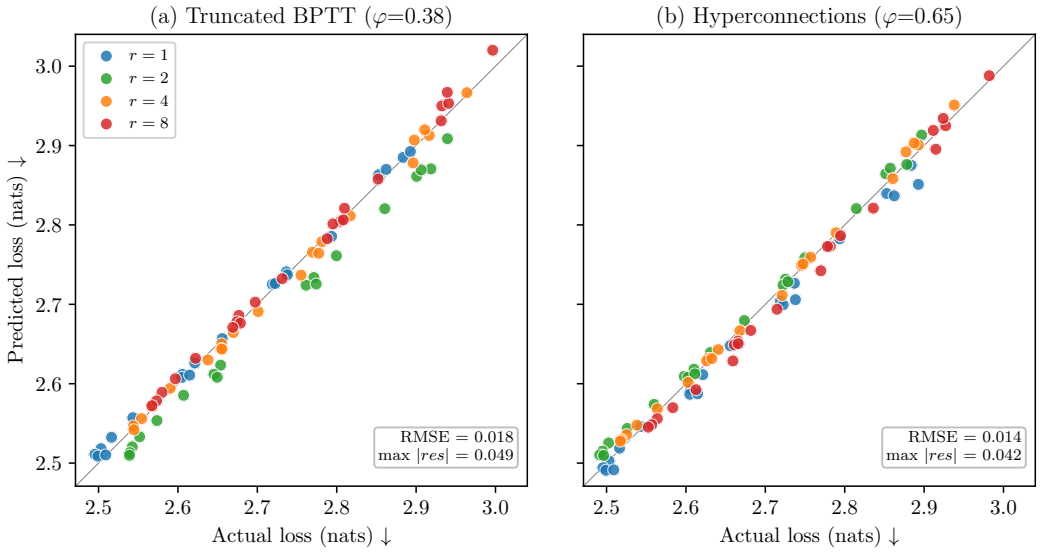


Figure 11: Case-study fit quality: predicted vs. actual validation loss under the joint-law refits of Table 3. Markers encode recurrence $r \in \{1, 2, 4, 8\}$ by colour. Annotation reports root-mean-square and maximum absolute residual.

Hyperconnections. φ rises from 0.45 to 0.65, so g_r at $r > 1$ grows and the compute-optimal width s^* contracts. The compute-optimal stars on the hc panel of Figure 5 (right) sit at smaller N^* than the corresponding full-BPTT linear-injection stars at matched budgets. Lower N^* at the same compute also means lower per-token inference FLOPs, the converse of the truncated BPTT result.

I Downstream Task Evaluation

We evaluate the trained models on downstream tasks to determine how the validation-loss gap between looped and non-looped models transfers across task types. We test whether looped LMs lose more on tasks that rely on information stored in their parameters, where their lower unique parameter counts may matter most, and whether they gain on tasks involving context use and multi-step reasoning. We study these questions across the baseline iso-FLOPs grid (Section 4), the truncated-BPTT and hyperconnection case studies (Section 5), and our s=34 extrapolation run (Section 4.2).

I.1 Evaluation Protocol

Our evaluation suite combines 18 tasks from the CORE benchmark [41] and Saunshi et al. [12] into four categories (Table 9). We retain the source benchmarks' few-shot settings, except that we reduce CoQA to 1-shot to fit our 2,048-token context window. Following Heineman et al. [42], we report per-token loss on the gold continuation and average the resulting task losses within each category. For compute-optimal comparisons, we select the checkpoint with the lowest validation loss for each architecture and compute budget.

I.2 Baseline Grid Results

We first examine how downstream performance changes with recurrence across the baseline iso-FLOPs grid of Section 4. Figure 12 reports average continuation loss for the compute-optimal model of each architecture at every compute budget.

Parametric knowledge follows the validation-loss ordering most clearly. The non-looped baseline performs best at every compute budget, and the gap generally grows with recurrence. Reading comprehension shows a different pattern: r=2 and r=4 remain competitive with the non-looped baseline, while r=8 consistently trails. Math word problems and reasoning primitives improve with compute but show no consistent ordering across recurrence counts.

Table 9: Downstream evaluation suite. LM and MC denote language-modelling and multiple-choice tasks. Samples gives the number of evaluated examples. TriviaQA and BigBench QA-Wikidata are uniformly subsampled to 10,000 examples.

Axis	Task	Samples	Shots	Type
Parametric knowledge *(closed-book factual questions)*	TriviaQA [43]	10,000	5	LM
	NaturalQuestions [44]	3,610	5	LM
	WebQuestions [45]	2,032	5	LM
	BigBench QA-Wikidata [46]	10,000	10	LM
	ARC-Easy [47]	2,376	10	MC
Reading comprehension *(answers determined by a passage in the prompt)*	Lambada-OpenAI [48]	5,153	0	LM
	TyDiQA-GoldP [49]	440	3	LM
	SQuADv2 [50]	5,928	3	LM
	DROP [51]	9,535	3	LM
	CoQA [52]	7,983	1	LM
Math word problems *(elementary-school arithmetic in words)*	SVAMP [53]	300	5	LM
	ASDiv [54]	2,305	5	LM
	MAWPS [55]	1,772	5	LM
Reasoning primitives *(variable look-ups and string puzzles)*	VarAssign d0 (math) [56]	1,000	5	LM
	VarAssign d0 (code) [56]	1,000	5	LM
	VarAssign d1 (math) [56]	1,000	5	LM
	VarAssign d1 (code) [56]	1,000	5	LM
	BigBench CS-algorithms [46]	1,320	10	LM

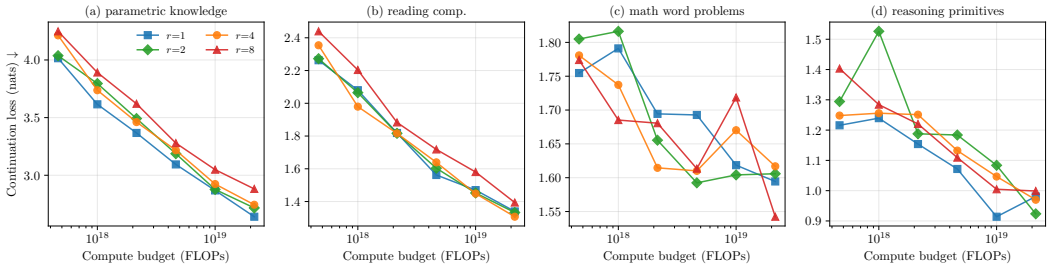


Figure 12: Downstream continuation loss across the baseline iso-FLOPs grid. Each point shows the compute-optimal model of one architecture at the corresponding compute budget. Lower is better.

To determine whether the results in Figure 12 exceed run-to-run variation, we repeat training the compute-optimal model of each architecture at the largest budget ($C{=}2.15{\times}10^{19}$ FLOPs) with three seeds. Table 10 reports the resulting means and standard deviations. The differences in parametric knowledge and reading comprehension are larger than the variation across seeds and reproduce the patterns observed across the grid. The differences on math word problems and reasoning primitives remain within run-to-run variation across training seeds.

Table 10: Results at the compute optimum for $C{=}2.15{\times}10^{19}$ FLOPs. The first row reports validation loss and the remaining rows report downstream continuation loss. Values are means and standard deviations across three training seeds. The final column reports φ fitted over the full grid, with 95% bootstrap intervals. "Too noisy" indicates that the task category does not admit a reliable fit.

	Loss (nats) ↓				
	$r{=}1$	$r{=}2$	$r{=}4$	$r{=}8$	φ
Validation loss	2.311 ± 0.001	2.340 ± 0.001	2.362 ± 0.002	2.403 ± 0.003	$0.46\ [0.41, 0.53]$
Parametric knowledge	2.653 ± 0.012	2.720 ± 0.003	2.754 ± 0.009	2.846 ± 0.032	$0.58\ [0.50, 0.70]$
Reading comprehension	1.352 ± 0.011	1.362 ± 0.012	1.355 ± 0.015	1.394 ± 0.016	$0.71\ [0.55, 0.85]$
Math word problems	1.619 ± 0.035	1.601 ± 0.014	1.605 ± 0.013	1.579 ± 0.032	too noisy
Reasoning primitives	0.963 ± 0.029	0.925 ± 0.041	0.940 ± 0.051	0.958 ± 0.035	too noisy

Table 11: Recurrence equivalence exponent φ measured from validation loss and downstream continuation loss. Truncated BPTT lowers φ relative to the baseline, while hyperconnections raise it.

Target	Baseline	Truncated BPTT	Hyperconnections
Validation loss	0.46	0.38	0.65
Parametric knowledge	0.58	0.50	0.80
Reading comprehension	0.71	0.67	0.94

Finally, we test whether the recurrence equivalence exponent measured from validation loss agrees with downstream performance (Table 10). We refit the joint scaling law using each task-category loss instead of validation loss. Parametric knowledge gives φ=0.58 and reading comprehension gives φ=0.71, compared with φ=0.46 for validation loss. These estimates preserve the main conclusion that recurrences provide only partial parameter equivalence, while their higher values indicate a smaller parameter-sharing cost on these downstream categories. The more challenging task categories, math word problems and reasoning primitives, do not admit reliable fits at our scale.

I.3 Case Study Results

We next test whether the effects of truncated BPTT and hyperconnections transfer to downstream performance. As in the baseline analysis, we obtain reliable fits only for parametric knowledge and reading comprehension. Table 11 compares the resulting recurrence equivalence exponents with those measured from validation loss. The downstream results preserve the ordering measured from validation loss. Truncated BPTT reduces the value of recurrence, while hyperconnections increase it. This agreement supports $\Delta\varphi$ as a diagnostic of the two case studies beyond validation loss.

I.4 Extrapolation Run Results

Finally, we evaluate the r=1 and r=4 extrapolation runs from Section 4.2. Both models have width s=34 and are trained on 47B tokens, corresponding to $\sim 4\times 10^{20}$ FLOPs and $\sim$20$\times$ the largest budget in the baseline grid. The runs are matched by training tokens, giving the looped model a $\sim$3% training-FLOPs advantage due to its input-injection overhead. The validation-loss gap is 0.061 nats, within the $[0.05, 0.08]$ nat range observed for r=4 across the baseline grid. Parametric knowledge again shows the largest downstream gap, while reasoning primitives remain approximately equal between the two models. Reading comprehension and math word problems show smaller gaps. Results are from a single training run, so we cannot estimate run-to-run variation at this scale.

Table 12: Validation and downstream losses for the s=34 extrapolation runs trained on 47B tokens. The gap is r=4 minus r=1, so positive values indicate that the looped model performs worse.

	r=1	r=4	Gap
Validation loss	2.047	2.108	+0.061
Parametric knowledge	2.123	2.204	+0.081
Reading comprehension	0.980	1.030	+0.050
Math word problems	1.494	1.522	+0.028
Reasoning primitives	0.802	0.797	−0.005

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The three main claims ($\varphi = 0.46$ recurrence-equivalence exponent, the iso-FLOPs gap at matched compute, and $\Delta\varphi$ separating recipe gains from architectural gains) are each supported by the joint-law fit and probe results, with no claim extending beyond the measured range of r, N, and D.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We added a dedicated limitations paragraph in Section 6 and reflect limitations throughout the paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper is empirical. The joint law is fit to data and interpreted as a local approximation, not derived as a theorem.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The architecture, FLOPs accounting and joint-law form are specified in Section 3, with many more details in the appendix. The pretraining corpus and tokenizer are public, and the code is available at `https://github.com/kschwethelm/looped-lm-scaling`.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The full code is added to the supplementary ZIP, including the training and evaluation scripts, the SLURM submission scripts that produced the results, and a README with setup and reproduction instructions. The pretraining corpus and tokenizer are public.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The training and validation recipe is summarised in Section 4.1, with the full architecture and initialisation in Appendix C. We show the results of our learning rate sweep and hyperparameter transfer validation in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The headline φ estimate is reported with a 95% block-bootstrap confidence interval over (budget, architecture) cells (see Appendix G.2). Stability is cross-checked by refitting on low- and high-budget halves of the grid.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report hardware and approximate total compute in Appendix B. Per-run compute is the iso-FLOPs budget of each grid cell, listed in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is foundational scaling-law research on a public pretraining corpus and small open-recipe transformers, with no human subjects, personal data, or model release. The authors have reviewed and conform to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: The work is foundational scaling-law research on a public pretraining corpus and small open-recipe transformers. There is no direct path to specific positive or negative societal applications beyond those generally associated with language model pretraining.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: No new datasets or model checkpoints are released.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All upstream code and datasets are cited in the paper, and used under their original licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new datasets or model checkpoints are released. The supplementary ZIP contains only reproduction code.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper involves no crowdsourcing and no human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper involves no human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used only as writing and coding aids and are not part of the methodology, experiments, or analysis.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.