More Than Parameter Count: Depth–Width Shape in Controlled Pre-LN Transformer Sweeps

Anonymous authors
Paper under double-blind review

Abstract

Parameter-count scaling models summarize architecture using a single quantity, N, even though models with similar parameter counts can allocate capacity very differently between depth and width. We ask whether depth–width shape adds predictive information in controlled Pre-LN Transformer sweeps. We analyze 30 decoder-only runs spanning 27M to 6.9B parameters and up to 143B training tokens. All runs use the same corpus, predetermined training seed, and final-step training-loss endpoint; within the documented 1B, 3B, and 7B tiers, the optimization configuration is held fixed while depth and width vary. The objective is to compare architecture shapes under controlled tier-specific protocols rather than after architecture-specific tuning. We compare a parameter-count baseline, a smooth D/W correction, and two thresholded excess-depth models. AICc favors the parameter-count baseline ($R^2 = 0.966$), whereas leave-one-run-out cross-validation within the observed architecture grid favors a logarithmic excess-depth model, reducing RMSE from 0.108 to 0.102 nats. Descriptively, the deepest configurations have higher final training loss than shallower, wider configurations by 0.157, 0.162, and 0.119 nats at the 1B, 3B, and 7B tiers. The 1B contrast is closely matched; the 3B contrast is approximately compute-matched despite small offsetting parameter and token differences; and the 7B contrast gives the wider run an advantage on all recorded budget dimensions. The fitted logarithmic onset lies at approximately 40–54 layers over $W = 512$–4096, but marks activation of a shape penalty rather than an optimal depth. The result is a modest, protocol-conditioned association and does not estimate seed-to-seed variability, validation performance, or a universal law of Transformer depth.

1 Introduction

Scaling laws enable an efficient way to budget large-model training. For a wide range of model sizes and corresponding data sizes, a few easily measurable variables provide good estimates of the final loss, mainly the number of parameters and the amount of training data (Kaplan et al., 2020; Hoffmann et al., 2022). These scaling principles have influenced the design and training of model families including GPT-3 (Brown et al., 2020), PaLM (Chowdhery et al., 2022), and LLaMA (Touvron et al., 2023). In many such fits, architecture is summarized by total parameter count N, while training data are summarized by the token budget T.

This compression is convenient but potentially lossy. For example, place two transformers of different shapes side by side (one with 10 layers at width 8,192, and one with 80 layers at width 2,900): their parameter counts can be approximately equivalent. A loss model defined only over N and T must therefore assign the same architectural contribution to both models. Modern language models frequently use deep stacks, but a parameter-only scaling model cannot determine whether additional depth is beneficial relative to additional width at a similar overall scale. In particular, it cannot represent whether reallocating capacity toward another layer, rather than additional width, changes the expected loss.

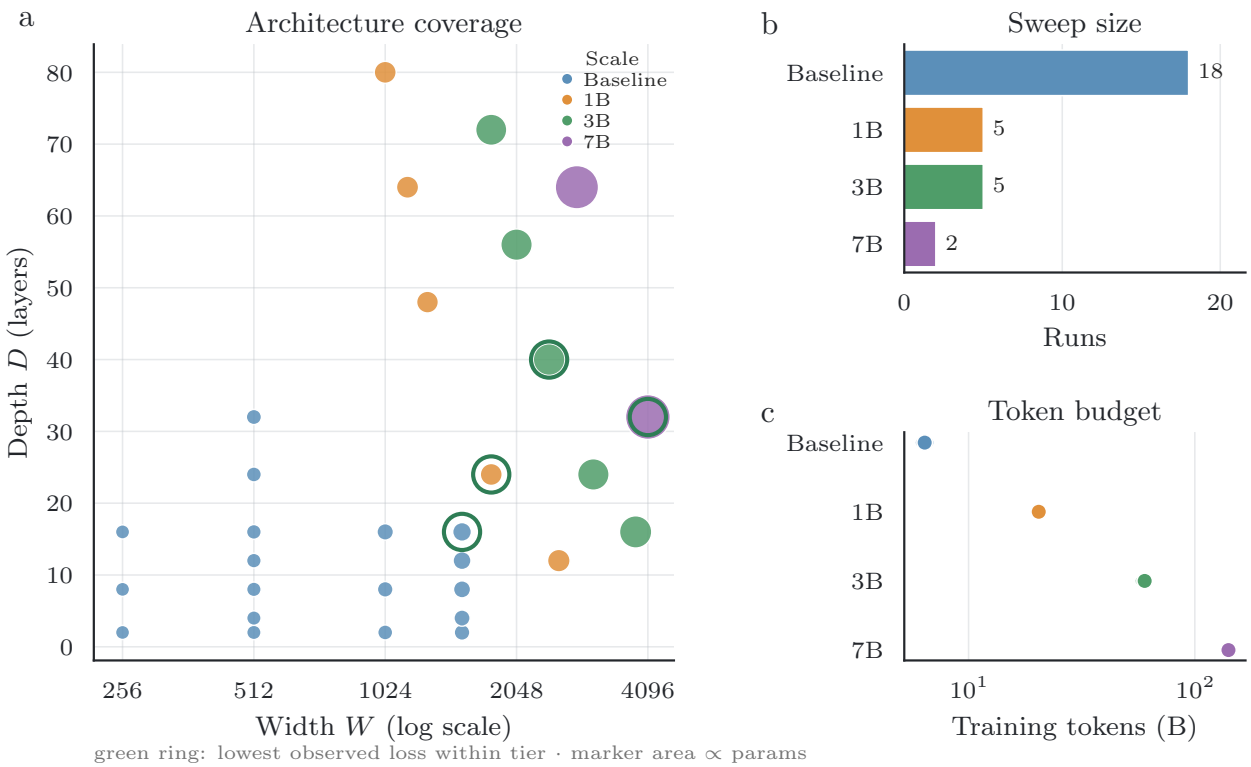

Figure 1: **Training grid and budgets. (a)** The 30 runs sit on the width–depth plane (marker area ∝ parameter count, colour = scale tier); a green ring marks the lowest loss observed within each predefined scale tier. The baseline tier gets a dense fixed-width sweep; the larger tiers trade density for approximate scale matching across widely separated depth–width shapes. **(b)** Runs per scale. **(c)** Training-token budget per scale (dot = median, bar = range; log axis).

We study this question under a deliberately controlled scope. The analysis comprises 30 completed training runs spanning 27M to 6.9B parameters. All used the same non-μP, decoder-only Pre-LN model family and SlimPajama corpus (Soboleva et al., 2023) (Figure 1). For the documented 1B, 3B, and 7B grids, batch size, sequence length, peak learning rate, and warmup are held fixed within tier while depth and width vary; complete matching batch-size and sequence-length metadata are unavailable for the baseline grid. We are not attempting a general theory of how many layers a Transformer should contain. Our question is far more specific: do loss functions that represent depth and width separately improve leave-one-run-out prediction within the observed architecture grid relative to a parameter-count baseline? The parameter-count baseline captures most of the loss variation across the runs; however, all the large-scale tiers show a consistent feature that an N-only model cannot represent: at each of the three largest scales, the deepest configuration does not produce the lowest observed loss.

We report the small-magnitude signal with its checks included. The thresholded excess-depth term improves leave-one-run-out prediction only to a modest extent, and the direct comparisons at 1B, 3B, and 7B differ by 0.157, 0.162, and 0.119 nats. Model selection is therefore done with AICc, leave-one-run-out cross-validation, run-table bootstrap sensitivity, and a sensitivity refit without the 7B runs. The interpretation stops where the data stop. One predetermined seed, one final-step loss definition, and fixed within-tier optimization set up the conditional comparisons; how the ordering would move across seeds, under architecture-specific transfer or depth-stabilized recipes, or on downstream evaluations is not estimated here.

1.1 Heuristic motivation for a thresholded term

If gradients attenuate on the way back through a deep residual stack, layers far from the loss get weak updates. As a heuristic for the backward signal through a D-layer residual network,

$$\|\nabla_\ell L\| \;\approx\; \|\nabla_D L\| \exp\!\left(-\frac{D-\ell}{\tau(W)}\right), \tag{1}$$

where $\tau(W)$ is a width-dependent persistence length. Depths beyond that point would leave the early layers weakly trained; those layers still add parameters and compute even as their marginal contribution to loss reduction may shrink.

2 Related Work

Neural scaling laws. Power-law improvements with scale precede the current era of language models (Hestness et al., 2017). Kaplan et al. (2020) turned them into an effective forecasting method, estimated $L \propto N^{-0.076}$, and found that the relationship was only weakly dependent upon the depth–width aspect ratio over its range. Chinchilla-type analyses then shifted focus toward the split between parameters and tokens (Hoffmann et al., 2022), and subsequent studies tested that prescription using various types of robustness testing (Schaeffer et al., 2025), cost accounting for inference (Sardana et al., 2023), and repeated-data scenarios (Muennighoff et al., 2023; Hernandez et al., 2022; Lovelace et al., 2026). Power laws have also been derived from the internal structure of data and features (Maloney et al., 2022), and extrapolation tests may alter how reliable the fitted exponents appear (Alabdulmohsin et al., 2022). In this literature, architecture is commonly summarized primarily through N, leaving depth and width implicit. The two closest references to our question are the aspect-ratio check of Kaplan et al. and the depth–width analysis of Alabdulmohsin et al. (2023) on compute-optimal vision transformers.

Depth, width, and expressivity. The depth–width trade-off is a longstanding question in neural-network theory and practice (Lu et al., 2017; Vardi et al., 2022; Zagoruyko & Komodakis, 2016). For fully connected networks, depth-separation results give evidence that bounded-depth networks can need exponentially large widths to represent certain classes of functions (Lu et al., 2017); the corresponding theoretical evidence for width separation is less extensive (Vardi et al., 2022). Convolutional-network results provide a complementary perspective, where sufficiently wide residual networks have shown the potential to match much deeper ResNets (Zagoruyko & Komodakis, 2016; He et al., 2016). The situation is once again mixed for transformers: some work demonstrates that different tasks favor depth or width in varying proportions (Tay et al., 2021), and other work identifies a depth-efficiency threshold, logarithmic in width, beyond which added depth stops paying off in expressive power (Levine et al., 2020). All of these studies concern expressivity or task accuracy. Our dependent variable is narrower, the final training loss of a language model under controlled tier-specific protocols.

Depth in transformer language models. Depth has been shown to help transformers, but the evidence does not suggest unbounded returns. With parameters kept constant, Petty et al. (2024) show rapidly saturating gains on compositional generalization, and Csordás et al. (2025) indicate that deeper LLMs do not produce many new operations in their depths; instead they mostly divide similar computations across an increasing number of layers. Both findings are compatible with diminishing marginal returns to depth, although neither directly establishes the training-loss relationship studied here. Layer-pruning studies (Sajjad et al., 2023; He et al., 2024; Kim et al., 2024), the looped or depth-like nature of some computation (Saunshi et al., 2025; Raposo et al., 2024), and bounds on bounded-depth self-attention (Hahn, 2020) are broadly compatible with the possibility of diminishing or redundant depth. We use this background to choose candidate loss terms, not to infer a mechanism.

Parameterization and hyperparameter transfer. The choice of shape comparison and the choice of optimization are inextricably linked. Maximal-update parameterization (μP) was introduced so that hyperparameters transfer across widths (Yang et al., 2021); unit scaling (Blake et al., 2024), alternative

parameterizations (Everett et al., 2024), different transfer mechanisms (Ghosh et al., 2025), and depth-wise transfer (Wu et al., 2026) all extend the same program. Our experiment intentionally does not use architecture-specific hyperparameter transfer or depth-dependent stabilization. The estimand is the relative behavior of architecture shapes within controlled tier-specific protocols, not the best performance obtainable after tuning each architecture separately. A μP or depth-aware replication would test whether the observed ordering transfers to a different optimization regime (Section 5.4).

Normalization, initialization, and training stability. Whether deeper transformers train well depends in part on normalization and initialization. Pre-LN reduces gradient instability at initialization compared to Post-LN (Xiong et al., 2020), residual-branch amplification can destabilize training (Liu et al., 2020), and the optimal placement of layer normalization has yet to be settled (Takase et al., 2022; Kim et al., 2025). DeepNet (Wang et al., 2022), ReZero (Bachlechner et al., 2021), and mean-field initialization analyses (Schoenholz et al., 2016) were all designed to stabilize training at large depths. Our research asks an alternative question: under documented tier-specific Pre-LN protocols without depth-specific stabilization, does final training loss retain a systematic residual dependence on depth–width shape after overall scale is represented? Any shape residual estimated here is conditional on these protocols and may change under a parameterization designed specifically for depth transfer.

Signal propagation and non-monotonic loss. The candidate terms draw a single qualitative idea from signal-propagation theory: the depth range supported by a shared recipe may increase with width. Stacked self-attention appears to favor rank collapse of representations (Noci et al., 2022); pure attention models tend to quickly lose rank as a function of the number of layers unless masks or layer normalization intervene (Dong et al., 2021; Wu et al., 2024); attention maps are quite variable across different layers and heads (Clark et al., 2019); and very recently there have been theoretical studies identifying forms of gradient failure at model initialization (Giorlandino et al., 2025). Since we log no per-layer gradients, we cannot assert a gradient-flow law. At a broad qualitative level, the non-monotonicity resembles capacity-axis effects studied in other neural-network regimes (Jacot et al., 2018; Yang & Littwin, 2021; Adlam & Pennington, 2020; Lee & Cherkassky, 2024), although the present comparison concerns finite Transformer shape under a roughly fixed scale budget.

3 Methods

Throughout the analysis, a proposed loss formula is treated as a predictive model and judged on leave-one-run-out prediction within the observed grid. Neither the compute argument nor the fitted penalty threshold is interpreted as an estimate of the loss-minimizing depth. A thresholded term is emphasized only when it improves LOOCV relative to the simpler baselines. Reported parameter estimates come from final fits on all 30 runs.

3.1 Dataset and training recipe

Our data are 30 completed Transformer runs, all trained with the released JAX/Flax implementation and its per-scale configurations. The grid is unbalanced on purpose. Most of it is small: eighteen baseline runs cover depths 2–32 and widths 256–1536 (27M–608M parameters), and seven of them form a depth sweep from 2 through 32 layers at a fixed width of $W = 512$, which the budget at that scale allowed. Five runs probe the 1B scale and five more the 3B scale, in each case spreading widely separated depth–width allocations across a narrowly matched scale or compute tier. Two runs form a 7B pair, a looser extension of the same idea. Some architectures were made deliberately deep because we wanted to know whether the tier-specific protocol kept rewarding capacity moved into depth. For every run we recorded the loss value, parameter count, tokens processed, and hardware metadata. Table 1 summarizes the inventory, and Appendix C lists each run.

Training recipe. All models are decoder-only Transformers in JAX/Flax with Pre-LN blocks, learned absolute positional embeddings, and GELU feed-forward layers. Training uses AdamW (Loshchilov & Hutter, 2017) with cosine learning-rate decay and no dropout. The corpus, the training seed, the loss-recording convention, and the model-family settings never change across the grid, and within each documented tier (1B,

Table 1: Experiment inventory. The final column is the lowest observed final loss within each predefined scale tier.

Scale	Runs	Parameter range	Token range	Lowest-loss observed shape	Final loss
baseline	18	27.4–608.3M	6.1–6.7B	16L/1536W	3.049
1B	5	1075.2–1206.4M	19.6–20.9B	24L/1792W	2.821
3B	5	2958.8–3225.2M	57.0–62.2B	40L/2432W	2.519
7B	2	6379.7–6863.1M	138.6–143.1B	32L/4096W	2.298

3B, 7B) the alternatives differ in depth and width only: batch size, sequence length, peak learning rate, and warmup are held constant (Table 9, Appendix B). What the recipe leaves out matters as much as what it fixes. There is no μP-style hyperparameter transfer (Yang et al., 2021), no DeepNet-style depth stabilization (Wang et al., 2022), and no depth-dependent residual or output scaling of any kind, $1/\sqrt{D}$ included. We held optimization fixed within each tier precisely so that per-architecture tuning never becomes part of the design. The price is that we can only say how one tier-specific protocol treats different depth–width shapes. Which shape would win after separate tuning is a different experiment.

Token and compute budgets. Token budgets grow with scale. A baseline run sees roughly 6.4B tokens (the observed range is 6.1–6.7B), a 1B run about 20B, and a 3B run about 60B. Depths and widths come in discrete steps, which rules out exact token matching; the tiers are therefore narrow instead of exact, and any budget difference that survives the design is reported as is. The 7B pair is the weakest match. Its deep and wide members trained on 138.6B and 143.1B tokens, and the shallower, wider model also carries more parameters and more estimated compute, so we treat this pair as directional evidence only, without attributing its loss difference to shape.

Measured outcome. Every run is scored by its *final training loss*, defined as the pre-update mean token-level cross-entropy on the final global batch under one shared logging convention. We chose this endpoint because it measures how well the tier protocol optimized the common pretraining objective while involving no architecture-dependent checkpoint selection. It also sets a hard limit on what the study can say. Intermediate curves and checkpoints were not kept, so validation loss, downstream behavior, and the path the loss took during training cannot be reconstructed or compared.

3.2 Candidate loss models

Four candidate models enter the comparison. Here and below, $\tilde{N} = N/10^9$ is the parameter count in billions and L is the final training loss:

$$M_0: \quad L = c + A\,\tilde{N}^{-\alpha}, \tag{2}$$

$$M_1: \quad L = c + A\,\tilde{N}^{-\alpha} + B\,(D/W)^{\beta}, \tag{3}$$

$$M_2: \quad L = c + A\,\tilde{N}^{-\alpha} + B\left[\max\Big(0, \tfrac{D}{c_d\,W^a} - 1\Big)\right]^q, \tag{4}$$

$$M_3: \quad L = c + A\,\tilde{N}^{-\alpha} + B\left[\max\big(0, \tfrac{D}{\kappa \ln W} - 1\big)\right]^q. \tag{5}$$

The baseline M_0 sees only parameter count. M_1 adds the smooth ratio D/W on top of it. The remaining two are threshold models: each adds an excess-depth penalty that stays at zero until depth crosses a width-dependent onset $D_{\text{crit}}(W)$, which M_2 writes as the power law $c_d W^a$ and M_3 as the local logarithmic form $\kappa \ln W$ (all logs are natural). Two warnings apply to the onset. It marks the point at which the penalty term turns on, which is not the same thing as the loss-minimizing depth, and in the case of M_3 it describes only the widths we actually studied, with no asymptotic content. Notation is collected in Table 4 (Appendix A).

3.3 Selection protocol

Each candidate was fit to all 30 runs by bounded nonlinear least squares, and every fit then went through the same four checks. The first is AICc (Hurvich & Tsai, 1989), computed on the residual sum of squares. The

Table 2: Comparison of candidate loss models. Each row fits one loss model to all 30 runs; lower AICc and LOOCV RMSE are better, and ΔLOOCV is the leave-one-out RMSE reduction over the parameter-count baseline (positive favours the row). We bold the LOOCV-preferred candidate only where it improves upon both the parameter-count and shape-ratio baselines under LOOCV; a paired resampling interval over held-out runs places its gap at +0.006 nats (95% interval $[0.0002, 0.013]$, favoured in 98% of resamples). AICc nevertheless favors M_0. The power-threshold scale reaches the boundary of its specified search range and is therefore not interpreted.

Model	Equation	AICc	LOOCV RMSE	ΔLOOCV	Fit RMSE	R^2
Parameter-only	$L = c + A\tilde{N}^{-\alpha}$	-133.4	0.108	–	0.096	0.966
Shape-ratio	$L = c + A\tilde{N}^{-\alpha} + B(D/W)^{\beta}$	-129.2	0.111	-0.002	0.094	0.967
Power excess-depth onset	$L = c + A\tilde{N}^{-\alpha} + B[\max(0, D/(c_d W^a) - 1)]^q$	-125.7	0.105	+0.004	0.089	0.970
Log excess-depth onset	$L = c + A\tilde{N}^{-\alpha} + B[\max(0, D/(\kappa \ln W) - 1)]^q$	-129.2	0.102	+0.006	0.089	0.970

second is leave-one-out cross-validation (LOOCV) RMSE, for which the model is refit on the remaining 29 runs before predicting each held-out point. The third is a run-table bootstrap with 160 resamples under a fixed seed; this probes how sensitive a fit is to the composition of the run table itself, not to seed-to-seed training noise. The last check refits everything without the two 7B runs. Our prespecified rule accepts a thresholded term only if its LOOCV beats the parameter-only baseline and the shape-ratio baseline at the same time. One caveat carries through all of this. The shape coefficients were constrained to $B \geq 0$ from the start, so the bootstrap intervals report how large B is under that constraint and say nothing on their own about its sign.

3.4 Compute constraints do not identify the depth–width split

Budgets computed on their own cannot differentiate among depth–width splits. With $N \approx 12DW^2$ and $C \approx 6NT$, a constrained budget limits $C \approx 72\, DW^2T$. If architecture affects loss only via N, equivalently via DW^2, then the first-order conditions in D and W are proportional to one another. Since the budget fixes the product DW^2T, it does not constrain how DW^2 is distributed across depth and width. A predictive model must therefore include an additional architecture-sensitive term if it is to distinguish depth–width allocations with the same value of DW^2; M_1–M_3 provide alternative forms of such a term. Appendix A provides the derivation.

4 Results

4.1 Depth–width loss geometry

The raw losses motivate a shape comparison before fitting any model (Figure 2). Parameter count accounts for the broad decline in loss, but within the larger tiers loss is not monotonic in depth. At 1B, the 80-layer run finishes at 2.978 nats and the lowest-loss observed 24-layer run at 2.821 nats, a difference of +0.157 nats. The corresponding 3B values are 2.681 at 72 layers and 2.519 at 40 layers, a +0.162-nat gap. The 7B pair separates by +0.119 nats: 2.417 for the 64-layer, width-2816 run against 2.298 for the 32-layer, width-4096 run. The fixed-width $W = 512$ baseline turns upward as well, from its lowest observed 3.435 nats at 16 layers to 3.467 nats at 24.

Exact matching is unreachable with discrete architecture choices, so the tiers are narrow rather than perfect. The closest contrast is at 1B, where the lower-loss 24L/1792W run has slightly fewer parameters and tokens than the 80L/1024W run (1.109 versus 1.112B parameters; 20.58 versus 20.90B tokens). At 3B the lower-loss 40L/2432W run carries 4.4% more parameters and 3.3% fewer tokens than the 72L/1792W run, with estimated training compute within roughly 1%, an approximately compute-matched pair. At 7B the shallower, wider configuration leads on parameters, tokens, and estimated compute alike, so that pair alone cannot pin the gap on shape rather than budget.

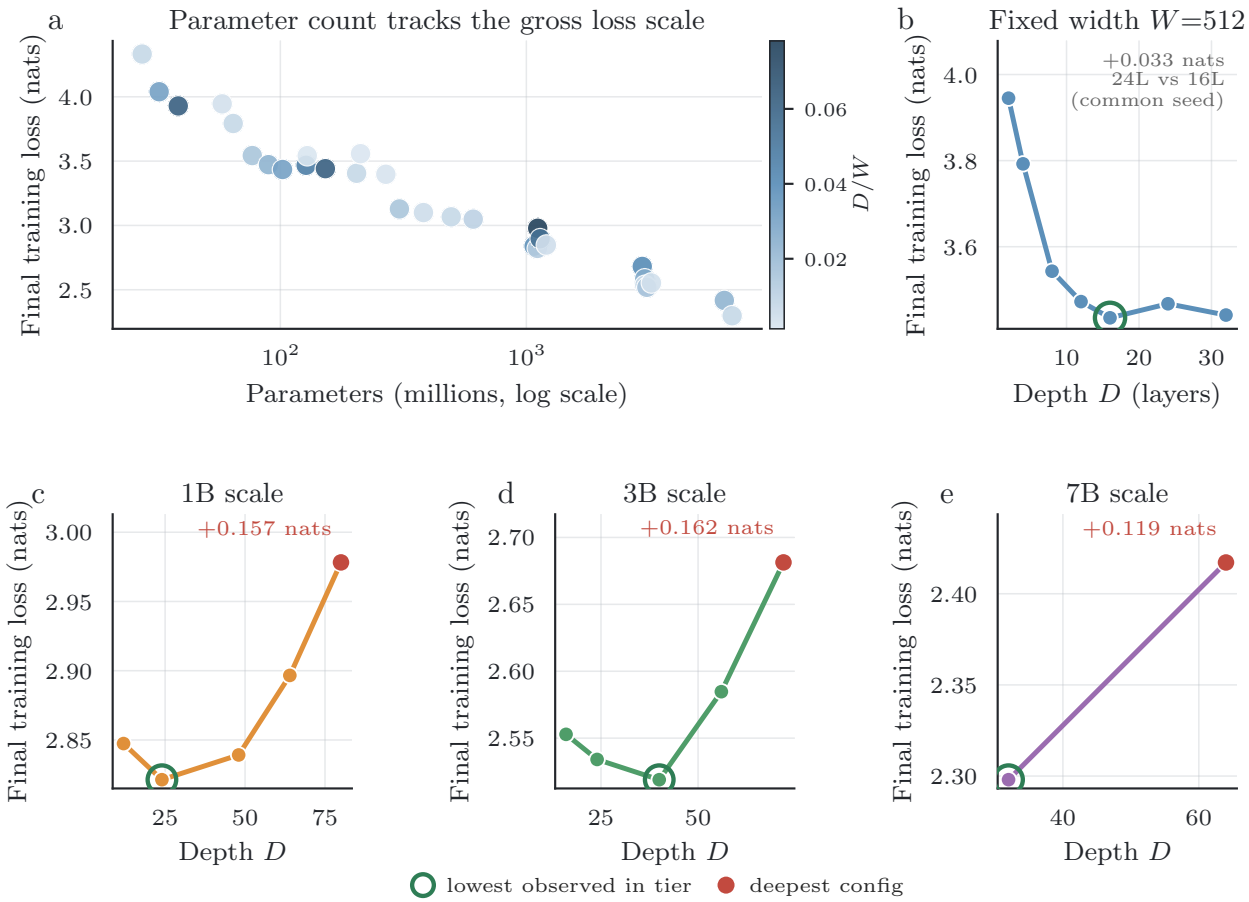

Figure 2: **Raw loss patterns by depth.** Panel **(a)** plots loss against parameter count on a log axis, with colour giving D/W. Panel **(b)** shows the fixed-width $W = 512$ baseline sweep; because the whole span is small, we annotate the rise past the lowest observed point (+0.033 nats between 24L and 16L on unrounded losses). Panels **(c–e)** give one depth profile per scale: a green ring marks the lowest loss observed in the tier, red marks its deepest member, and width co-varies at a roughly fixed budget. In each tier the deepest configuration lands above the tier's best run by the margin printed in the panel. Every panel uses the common-seed runs, the documented 1B–7B tiers were trained under controlled within-tier optimization settings, and cross-seed error bars are not available. Per-panel y-axes are independent.

4.2 Model comparison

Model preference changes with the selection criterion (Table 2, Figure 3). AICc favors the simpler parameter-count model: M_0 has $R^2 = 0.966$ and the lowest AICc of the four candidates (-133.4). Its effective scale exponent is $\alpha = 0.124$ with a bootstrap 95% interval of [0.092, 0.231], which excludes zero. This positive estimate tracks the campaign-wide decline in loss with scale. Parameter and token budgets co-vary across tiers, so its size is not a separately identified universal exponent, and it shifts with the fitting range, reaching 0.155 once the 7B runs are removed (Section 4.4).

How much a shape term helps depends on the form it takes. The smooth ratio does not help: M_1 lands at 0.111 LOOCV RMSE against 0.108 for M_0, and its exponent β runs into the search bound. The log-threshold model is the one that gains ground. M_3 brings LOOCV RMSE down to 0.102 and predicts 19 of the 30 held-out runs with smaller squared error; Table 2 reports the paired-resampling detail. The bootstrap, run under the prespecified $B \geq 0$ bound, puts B in [0.05, 0.97] and q in [0.28, 3.6] (Section 3.3).

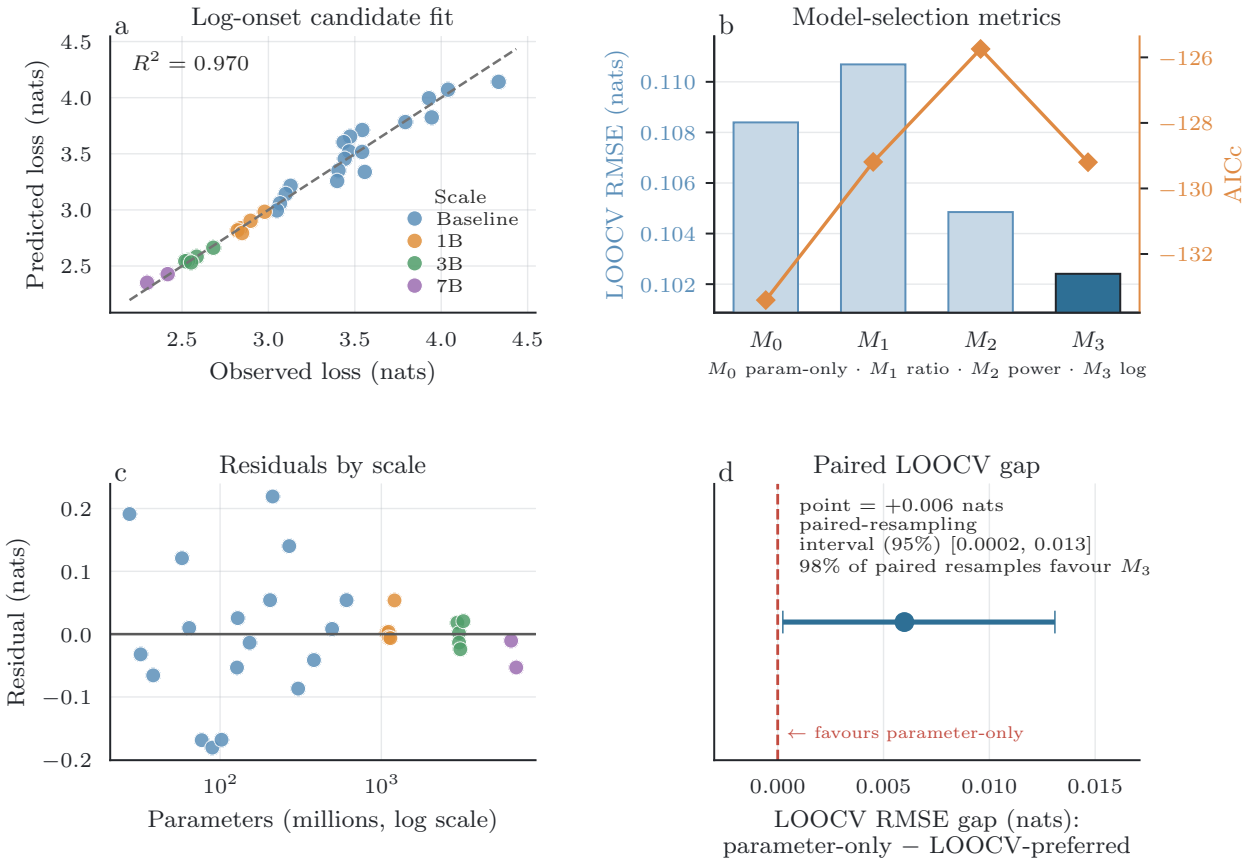

Figure 3: **Candidate-model comparison. (a)** Observed against predicted loss for the LOOCV-preferred candidate (M_3), with the 1:1 line. **(b)** LOOCV RMSE (bars; preferred candidate shaded) and AICc (diamonds, right axis) across candidates. **(c)** Residuals across scale show no systematic scale-dependent bias. **(d)** Paired leave-one-out RMSE gap (parameter-only − LOOCV-preferred): the paired resampling interval narrowly excludes zero yet stays small (+0.006 nats), while AICc still favors the parameter-only model.

Why do the two criteria disagree? At $n = 30$ the AICc penalty for two extra parameters is heavy, and M_0 keeps the best score. LOOCV asks about held-out prediction instead, and there M_3 cuts RMSE by roughly five percent. Set against the loss differences across scales this is a modest amount, so M_0 stays a perfectly reasonable parsimonious forecaster. The gain is stable, though: dropping any one of the three deepest large-tier runs leaves it intact (Section 4.4).

4.3 Boundary diagnostics for the threshold forms

The observed grid does not constrain the two threshold families equally well. For the log form, the fit gives $D_{\text{crit}}(W) = 6.48 \ln W$, and under the stated nonlinear fit and bounds κ sits interior to its search range ($\kappa = 6.48$, bootstrap [3.85, 8.18]; $q = 0.88$). The power form W^a fares worse. Its scale c_d climbs to the upper search bound, which makes M_2 boundary-limited. M_2 stays in the comparison because it was prespecified, but we read nothing into its threshold scale.

The fitted log form places the penalty onset at approximately 40–54 layers over $W = 512$–4096. Under this form, the penalty is active for the 48L, 64L, and 80L runs at 1B; the 56L and 72L runs at 3B; and the 64L run at 7B. It remains inactive for the lower-depth configurations and for the complete $W = 512$ baseline sweep. The small rise between 16 and 24 layers in the $W = 512$ sweep occurs below the fitted onset and is therefore not explained by the M_3 penalty. With M_2 boundary-limited, we do not extrapolate a power

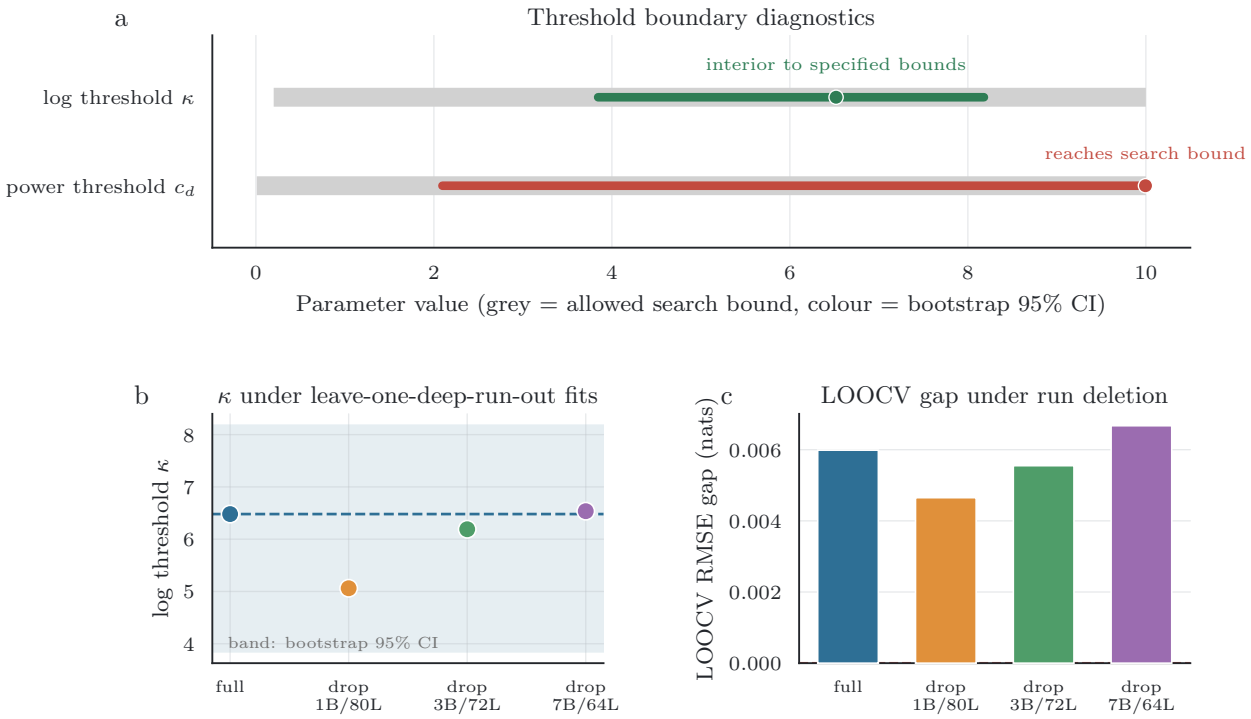

Figure 4: **Sensitivity of the log threshold.** **(a)** Bootstrap 95% intervals against the search bound: the interval for the log threshold κ remains interior to the specified bounds, whereas the power-law c_d reaches its search bound. **(b)** The refitted κ values remain within the full-data bootstrap interval when each deepest large-tier run is removed individually (band = full-data bootstrap 95% interval). **(c)** The parameter-only−LOOCV-preferred-candidate gap under the same deletions, consistent with Tables 2 and 7.

law to widths beyond those tested here. Table 8 (Appendix A.6) aligns the implied onset against the depths actually observed.

4.4 Sensitivity analyses

Excluding the two 7B runs does not change the ordering of the candidate models (Figure 4). If we refit M_3 excluding the 7B runs, we obtain $\kappa \approx 6.67$; this retains the same candidate ordering as before and keeps M_0's exponent positive ($\alpha = 0.16$). Removing each of the three deepest runs individually (Table 7), M_3 retains lower LOOCV RMSE than M_0 by +0.005 to +0.007 nats, with $\kappa \in [5.1, 6.5]$.

Token term and within-tier identification. There is an obvious concern that the depth term is proxying for something omitted from the analysis. A natural omitted-variable concern is token budget, which differs approximately twentyfold across tiers (Table 3). Two tests speak to this. Adding a token term $E\tilde{T}^{-\delta}$ gives no improvement in LOOCV over M_0 (0.116 versus 0.108); $\tilde{T}$ and $\tilde{N}$ are collinear across tiers, so the token term cannot be separately identified. When the token term is included, the estimated excess-depth coefficient is 0.22, with a central bootstrap interval of $[0.01, 1.56]$, under the prespecified nonnegative bound; κ stays near 6.35. A stronger test replaces the parameter-count baseline with per-tier intercepts, which capture all between-tier level differences including the confounded token budget. The LOOCV improvement associated with the excess-depth term therefore persists after between-tier level differences, including the tier-level N–T relationship, are absorbed by tier-specific intercepts. This supports a within-tier source for the predictive improvement, although it does not convert the observational architecture grid into a causal estimate of depth.

Table 3: Token-budget and within-tier sensitivity analyses (secondary analyses of the existing run table). *(a)* Adding a token term $E\tilde{T}^{-\delta}$ ($\tilde{T} = T/10^9$) does not improve leave-one-out prediction over parameter-only, since $\tilde{T}$ and $\tilde{N}$ are collinear across scale tiers and the token coefficient is not separately identified; the estimated excess-depth coefficient is 0.22 with a central bootstrap interval of $[0.01, 1.56]$ under the prespecified nonnegative bound, with the log threshold essentially unchanged ($\kappa = 6.35$). *(b)* Replacing the parameter-count model with per-tier intercepts (which absorb all between-scale level differences, including the confounded token budget) and adding the log-depth term still improves leave-one-out RMSE, by +0.010 nats. The predictive improvement remains after between-tier differences are absorbed; this supports a within-tier source without making a causal claim. ΔLOOCV is the reduction versus each panel's reference row (positive favours the row).

Model	k	Fit RMSE	LOOCV	ΔLOOCV	Excess-depth B
(a) Adding a token term $E\tilde{T}^{-\delta}$ (reference: M_0)					
Parameter-only (M_0)	3	0.096	0.108	–	–
+ token $E\tilde{T}^{-\delta}$	5	0.093	0.116	-0.007	not separately identified (E CI $[0.00, 3.68]$)
+ token + log-depth	8	0.085	0.108	+0.000	0.22 $[0.01, 1.56]$ (nonnegative bound imposed)
(b) Per-tier intercepts (reference: intercepts + parameter-count term)					
Tier intercepts + parameter-count term	6	0.090	0.113	–	–
+ log-depth	9	0.084	0.104	+0.010	0.18 $[0.02, 0.60]$ (nonnegative bound imposed)

4.5 Scope of supported claims

All claims are conditional on the design: shared seed, corpus, model family, and final-step loss, with optimization held fixed within the documented 1B, 3B, and 7B grids while depth and width vary. Within that scope, parameter count explains most of the loss variation and the excess-depth term adds a small LOOCV gain. Seed-to-seed variation, architecture-specific tuning, μP or depth-stabilized transfer, and downstream generalization each need their own experiment.

5 Discussion

5.1 Implication for scaling fits

Under the documented tier-specific, non-μP Pre-LN protocols, depth–width shape carries a small but detectable amount of predictive information beyond the parameter-count baseline; Section 4.1 details how tightly each tier contrast is matched. In the fitted models, the excess-depth term improves leave-one-run-out prediction by a modest margin, the parameter-count baseline still keeps the best AICc, and only the logarithmic threshold estimate stays interior to its specified search bounds.

A parameter-count-only law cannot represent this within-tier shape residual separately from N. That matters mostly when an N, T law is carried across architectural families rather than being used within one family. Anyone who currently performs depth sweeps can use the protocol described here to compare candidate shape terms without concluding from one architecture pairing alone. Similarly, anyone who trains using μP or a depth-stabilized recipe should interpret these runs as controlled tier-specific baselines, not as predictions about their own setting.

5.2 Why shape can be missed

One reason for the predictive success of parameter-count laws is that they suppress architectural detail. The trade-off is that a shape effect stays obscured until a grid reaches the region where that effect takes place. Kaplan et al. (2020) reported weak dependence of loss on aspect ratio, and the $W = 512$ sweep here shows a comparably small 0.033-nat rise from its lowest observed point. Every configuration in that sweep sits below the onset estimated by M_3, so the LOOCV-preferred candidate attributes none of that small-scale fluctuation to its excess-depth penalty; the fitted penalty is informed by the deeper runs in the larger tiers. A shallow sweep and a sweep well past the fitted onset can therefore disagree about whether an explicit shape term helps prediction, and the ordering may change under a different parameterization or tuning regime.

5.3 Interpreting the depth penalty

Gradient starvation supplied the thresholded functional form (Section 1.1), but these runs identify no mechanism: layer-wise gradients were never recorded during training, and rank collapse in deep self-attention (Noci et al., 2022; Dong et al., 2021), redundant computation spread through multiple layers (Csordás et al., 2025), and prunability at the layer level (Sajjad et al., 2023; He et al., 2024) are all broadly compatible with the same observed pattern. What the model comparisons show is limited to the shared observable: an excess-depth term improves leave-one-run-out prediction within this run set.

The boundary-limited power-law threshold is a caution against over-reading fitted forms. The logarithmic form places the penalty onset in the neighborhood of 40–54 layers over $W = 512$–4096; it does not license extrapolating a W^a law beyond the observed range.

A heuristic scoped to these protocols. $D_{\text{crit}}(W) \approx 6.48 \ln W$ comes from fitting the log model, corresponding to an excess-depth onset of approximately 40–54 layers over $W = 512$–4096. Architectures deeper than this onset carry a positive shape-residual penalty relative to the parameter-count baseline. The onset is not an estimate of the loss-minimizing depth: total predicted loss also depends on the parameter-count term, and the observed grid does not support continuous optimization over D and W. We therefore offer the fitted onset only as a protocol-conditioned hypothesis to test in future architecture sweeps.

5.4 Controlled design, scope, and future work

The campaign was designed as a controlled comparison. All architectures use the same Pre-LN block family, corpus, predetermined training seed, and final-step loss definition; within the documented 1B, 3B, and 7B tiers, the optimizer settings and schedule are held fixed. These shared components reduce variation unrelated to architecture shape and define the estimand of the study: the conditional ordering of depth–width configurations under controlled tier-specific protocols. These choices define the study's estimand rather than an attempt to optimize every architecture independently.

Controlled tier-specific estimand. The absence of μP, depth-dependent residual scaling, and per-architecture hyperparameter tuning is deliberate. The study asks whether shape matters when a documented tier-specific protocol is held fixed, as it often is in architecture sweeps and scaling-law campaigns. It does not ask which architecture would achieve the lowest loss after individual tuning. Methods such as μP, DeepNet, or ReZero may change the ordering, which would represent a different experimental regime rather than invalidate the conditional comparison reported here.

Common seed and unreplicated architectures. The architecture grid uses one predetermined training seed, providing a common stochastic protocol. Different tensor shapes mean that this is not a paired initialization of corresponding parameters. The design establishes an ordering for that seed but cannot quantify how the ordering varies across seeds. For example, it remains unknown whether the 0.033-nat difference in the $W = 512$ sweep would persist in independently seeded runs. The run-table bootstrap assesses sensitivity to the architectures included in the fit, not variation across repeated training runs.

Common final-step training-loss endpoint. We use final training loss as a common optimization outcome, avoiding architecture-dependent checkpoint selection and downstream evaluation choices. The conclusions therefore concern optimization of the shared language-modeling objective. Validation curves, intermediate checkpoints, and downstream evaluations were not retained, so the study cannot determine whether the same ordering applies to held-out loss, in-context learning, or task performance.

Approximate tier matching. The 1B and 3B grids were designed to compare substantially different depth–width allocations within narrow scale and compute ranges, but exact equality is unavailable because valid architecture dimensions are discrete. The 1B contrast is closely matched; the 3B contrast is approximately compute-matched despite offsetting parameter and token differences. At 7B, the lower-loss shallower/wider run has more parameters, tokens, and estimated compute, so that pair is treated as directional evidence rather than an isolated estimate of shape.

Domain and architecture scope. The fitted constants are conditional on decoder-only Pre-LN Transformers trained on SlimPajama. They may not transfer to code, mathematics, multilingual corpora, alternative positional encodings, Post-LN architectures, or depth-stabilized parameterizations.

The most informative future work is a new controlled campaign rather than further functional-form search on the same 30 outcomes: replicate the closely matched 1B and 3B architecture contrasts across several seeds; evaluate all runs on a common held-out token set; repeat the comparison under μP and depth-aware residual scaling; construct a more closely matched 7B pair; and record layer-wise gradients and intermediate checkpoints to distinguish among possible mechanisms.

6 Conclusion

We studied architecture-conditioned scaling in 30 controlled Pre-LN Transformer runs spanning 27M to 6.9B parameters. The corpus, predetermined training seed, and final-step loss definition are shared across the grid. Within the documented 1B, 3B, and 7B tiers, the optimization configuration is fixed while depth and width vary. A parameter-count baseline explains most cross-run loss variation and is preferred by AICc. A logarithmic excess-depth candidate nonetheless earns a small LOOCV improvement within the observed grid, and in each documented tier the deepest configuration finishes above a shallower, wider alternative (Section 4.1). How tightly the pairs are matched varies, from the close 1B contrast through the approximately compute-matched 3B pair to the 7B pair, where the wider run leads on every recorded budget dimension.

The practical conclusion is conditional, not universal. Before an N, T scaling model is carried across very different architecture shapes, a controlled within-tier sweep can check whether depth and width leave a predictable residual beyond parameter count. The fitted logarithmic onset marks activation of an additional shape term; it does not estimate an optimal depth or demonstrate a gradient-starvation mechanism. Whether the same ordering transfers across random seeds, validation outcomes, individually tuned recipes, μP, or depth-stabilized parameterizations remains an open empirical question.

Reproducibility Statement

We provide the training stack (JAX/Flax model and training loop, SlimPajama pipeline, per-scale configurations; Appendix B), as well as the results of every run in `training_runs.csv`. These materials are intended to reproduce the training procedure subject to hardware, input-pipeline, and accelerator nondeterminism (training seed 42). The deterministic `run_analysis.py` script (seed 20260619) regenerates the downstream analysis and `verify_model_comparison.py` reproduces the model-comparison table (Table 2) under the seeded bootstrap. The release records the analysis dependencies, per-run configurations and metadata, nonlinear-fit initialization and bounds (Table 6), the GPT-2 tokenizer, and the exact final-loss definition used here.

References

Ben Adlam and Jeffrey Pennington. The neural tangent kernel in high dimensions: Triple descent and a multi-scale theory of generalization. In *International Conference on Machine Learning*, 2020.

Ibrahim Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in language and vision. In *Advances in Neural Information Processing Systems*, 2022.

Ibrahim Alabdulmohsin, Xiaohua Zhai, Alexander Kolesnikov, and Lucas Beyer. Getting ViT in shape: Scaling laws for compute-optimal model design. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W Cottrell, and Julian McAuley. ReZero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, 2021.

Charlie Blake, Constantin Eichenberg, Josef Dean, Lukas Balles, Luke Y Prince, Björn Deiseroth, Andres Felipe Cruz-Salinas, Carlo Luschi, Samuel Weinbach, and Douglas Orr. u-μP: The unit-scaled maximal update parameterization. *arXiv preprint arXiv:2407.17465*, 2024.

Tom B Brown et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Aakanksha Chowdhery et al. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does BERT look at? An analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, 2019.

Róbert Csordás et al. Do language models use their depth efficiently? *arXiv preprint arXiv:2505.13898*, 2025.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pp. 2793–2803. PMLR, 2021.

Katie Everett et al. Scaling exponents across parameterizations and optimizers. In *International Conference on Machine Learning*, 2024.

Nikhil Ghosh et al. Understanding the mechanisms of fast hyperparameter transfer. *arXiv preprint arXiv:2512.22768*, 2025.

Alessio Giorlandino et al. Two failure modes of deep transformers and how to avoid them: A unified theory of signal propagation at initialisation. *arXiv preprint arXiv:2505.24333*, 2025.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Shwai He et al. What matters in Transformers? Not all attention is needed. *arXiv preprint arXiv:2406.15786*, 2024.

Danny Hernandez, Tom Brown, Tom Conerly, et al. Scaling laws and interpretability of learning from repeated data. *arXiv preprint arXiv:2205.10487*, 2022.

Joel Hestness et al. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Clifford M Hurvich and Chih-Ling Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, 1989.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018.

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

Bo-Kyeong Kim et al. Shortened LLaMA: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 2024.

Jeonghoon Kim et al. Peri-LN: Revisiting the normalization layer in the Transformer architecture. *arXiv preprint arXiv:2502.02732*, 2025.

Eng Hock Lee and Vladimir Cherkassky. Understanding double descent using a VC-theoretical framework. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

Yoav Levine et al. Limits to depth efficiencies of self-attention. In *Advances in Neural Information Processing Systems*, 2020.

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Justin Lovelace et al. Prescriptive scaling laws for data-constrained training. *arXiv preprint arXiv:2605.01640*, 2026.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, 2017.

Alexander Maloney, Daniel A Roberts, and James Sully. A solvable model of neural scaling laws. *arXiv preprint arXiv:2210.16859*, 2022.

Niklas Muennighoff, Alexander M Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. In *Advances in Neural Information Processing Systems*, 2023.

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. In *Advances in Neural Information Processing Systems*, 2022.

Jackson Petty, Sjoerd van Steenkiste, Ishita Dasgupta, Fei Sha, Dan Garrette, and Tal Linzen. The impact of depth on compositional generalization in transformer language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7239–7252, 2024.

David Raposo et al. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77, 2023.

Nikhil Sardana et al. Beyond Chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*, 2023.

Nikunj Saunshi et al. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations (ICLR)*, 2025.

Rylan Schaeffer et al. Evaluating the robustness of Chinchilla compute-optimal scaling. *arXiv preprint arXiv:2509.23963*, 2025.

Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Daria Soboleva, Faisal Al-Khateeb, et al. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. *Cerebras Systems*, 2023.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. On LayerNorm and residual connections in Transformers. *arXiv preprint arXiv:2206.00330*, 2022.

Yi Tay et al. Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*, 2021.

Hugo Touvron et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Gal Vardi, Gilad Yehudai, and Ohad Shamir. Width is less important than depth in ReLU neural networks. In *Conference on Learning Theory (COLT)*, 2022.

Hongyu Wang et al. DeepNet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.

Shenxi Wu et al. Hyperparameter transfer laws for non-recurrent multi-path neural networks. *arXiv preprint arXiv:2602.07494*, 2026.

Xinyi Wu et al. On the role of attention masks and layernorm in transformers. In *Advances in Neural Information Processing Systems*, 2024.

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

Greg Yang and Etai Littwin. Tensor Programs IIb: Architectural universality of neural tangent kernel training dynamics. In *International Conference on Machine Learning*, 2021.

Greg Yang, Edward J 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 *Advances in Neural Information Processing Systems*, volume 34, pp. 17084–17097, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

A Model Comparison Details

A.1 Notation

Table 4: Notation used in the candidate equations.

Symbol	Meaning
D	Transformer depth in layers
W	Hidden width
N, $\tilde{N}$	Parameter count, and parameter count in billions
T	Training-token count
L	Final training loss
α	Effective exponent of the parameter-count baseline in this campaign
D/W	Shape ratio used by the smooth baseline M_1
$D_{\mathrm{crit}}(W)$	Candidate onset of the excess-depth penalty; not an optimum or theorem
a, c_d	Power-law threshold parameters, $D_{\mathrm{crit}} = c_d W^a$
κ	Local natural-log onset parameter, $D_{\mathrm{crit}} = \kappa \ln W$
B, q	Excess-depth penalty strength and exponent

A.2 Selection statistics and definitions

Each candidate $f(\cdot\,;\theta)$ is fit to $n = 30$ runs by bounded nonlinear least squares using SciPy's trust-region reflective solver. For run i with architecture $\mathbf{x}_i = (D_i, W_i, N_i, T_i)$ and loss L_i,

$$\hat{\theta} \;=\; \underset{\theta \in [\theta_{\mathrm{lo}},\, \theta_{\mathrm{hi}}]}{\arg\min} \sum_{i=1}^{n} \big(L_i - f(\mathbf{x}_i;\theta)\big)^2, \tag{6}$$

with per-model box bounds listed in Table 6; power terms use floored bases $\max(\cdot, \varepsilon)$, $\varepsilon = 10^{-12}$. Each full-data and leave-one-out fit uses the stated single initialization and at most 12,000 function evaluations; bootstrap fits with non-finite predictions or solver failures are omitted from the corresponding resample summary. With fitted values $\hat{L}_i$, residuals $r_i = L_i - \hat{L}_i$, $\mathrm{SSE} = \sum_i r_i^2$, and k free mean-function parameters,

$$\mathrm{RMSE} = \sqrt{\mathrm{SSE}/n}, \qquad R^2 = 1 - \mathrm{SSE}\Big/\sum_i (L_i - \bar{L})^2, \qquad \mathrm{AICc} = n \ln \tfrac{\mathrm{SSE}}{n} + 2k + \frac{2k(k+1)}{n-k-1}. \tag{7}$$

Here, k counts the fitted mean-function parameters under the convention used by the analysis script. The small-sample correction in Equation (7) penalizes M_2 and M_3 heavily enough at $n = 30$ for AICc to prefer M_0 despite the better in-sample fit of the richer models.

Leave-one-out cross-validation. For each run i, refit on the other $n-1$ runs to obtain $\hat{\theta}_{-i}$ and predict $\hat{L}_{-i} = f(\mathbf{x}_i; \hat{\theta}_{-i})$, giving

$$\mathrm{RMSE}_{\mathrm{LOO}} = \sqrt{\tfrac{1}{n} \textstyle\sum_{i=1}^{n} \big(L_i - \hat{L}_{-i}\big)^2}. \tag{8}$$

Run-table bootstrap stability and boundary diagnostics. We refit $R_{\mathrm{boot}} = 160$ bootstrap resamples, sampled with replacement under a fixed seed. For parameter θ_j, the central 95% interval runs from the 2.5th to the 97.5th percentile. An estimate counts as interior when that interval sits strictly inside the search range; this is a boundary diagnostic, not an identifiability result. c_d in M_2 reaches its upper bound and is boundary-limited. Since B is constrained nonnegative, its positive sign is not a bootstrap discovery.

Paired leave-one-out gap. Held-out squared errors e_i^0 and e_i^3 for M_0 and M_3 are compared across 2000 paired resamples of the 30 held-out-run error pairs, giving the distribution of $\Delta = \sqrt{\frac{1}{n}\sum_i e_i^0} - \sqrt{\frac{1}{n}\sum_i e_i^3}$. Because the LOOCV training sets overlap, this percentile interval is interpreted as a descriptive paired stability analysis over the observed runs rather than as an independent-sample confidence interval.

Table 5: Candidate formula definitions. $\tilde{N}$ denotes parameters measured in billions.

Model	Equation	Role
Parameter-count baseline	$L = c + A\tilde{N}^{-\alpha}$	Parameter-count baseline
Shape-ratio	$L = c + A\tilde{N}^{-\alpha} + B(D/W)^{\beta}$	Smooth architecture term
Power excess-depth onset	$L = c + A\tilde{N}^{-\alpha} + B[\max(0, D/(c_d W^a) - 1)]^q$	Prespecified power-onset candidate
Log excess-depth onset	$L = c + A\tilde{N}^{-\alpha} + B[\max(0, D/(\kappa \ln W) - 1)]^q$	Prespecified logarithmic-onset candidate

Table 6: Nonlinear-fit settings for the prespecified candidates. Bounds are inclusive box bounds; $B \geq 0$ is imposed for each non-baseline shape term. All fits use SciPy `curve_fit` with the trust-region reflective method, one stated initialization, and at most 12,000 function evaluations.

Model	Parameters	Initialization	Lower/upper bounds
M_0	c, A, α	$(2.2, 1.8, 0.25)$	$(0, 0, 0.001)$ / $(10, 20, 2)$
M_1	c, A, α, B, β	$(2.1, 1.8, 0.25, 25, 1)$	$(0, 0, 0.001, 0, 0.001)$ / $(10, 20, 2, 10^5, 5)$
M_2	$c, A, \alpha, B, c_d, a, q$	$(2.1, 1.8, 0.25, 0.15, 0.95, 0.44, 1)$	$(0, 0, 0.001, 0, 0.01, 0.001, 0.25)$ / $(10, 20, 2, 10, 10, 1.5, 4)$
M_3	$c, A, \alpha, B, \kappa, q$	$(2.1, 1.8, 0.25, 0.15, 2.4, 1)$	$(0, 0, 0.001, 0, 0.2, 0.25)$ / $(10, 20, 2, 10, 10, 4)$

Excess-depth penalty. Thresholded terms use $\text{excess}(D, W) = \max(0, D/D_{\text{crit}}(W) - 1)$ and penalty $= B\,[\text{excess}(D, W)]^q$, zero for $D \leq D_{\text{crit}}(W)$.

A.3 Candidate formula definitions

Table 5 restates the four candidate models and the role each plays in the comparison.

A.4 Compute constraints do not identify the depth–width split

We give the explicit form of this argument, following Section 3.4. Write parameter count as $N \approx 12DW^2$ and training compute as $C \approx 6NT$, so that $C \approx 72\,DW^2T$. Consider any loss model whose architecture dependence enters only through N:

$$L(D, W, T) = \frac{A}{(12DW^2)^{\alpha}} + \frac{E}{T^{\delta}}. \tag{9}$$

Substituting the compute constraint $T = C/(72DW^2)$ reduces L to a function of the single combination $u \equiv DW^2$:

$$L = A\,(12u)^{-\alpha} + E\,(72u/C)^{\delta}. \tag{10}$$

Since L depends on D and W only through $u = DW^2$, the stationarity conditions satisfy

$$\frac{\partial L}{\partial D} = L'(u)\,W^2, \qquad \frac{\partial L}{\partial W} = L'(u)\,2DW, \tag{11}$$

both of which are proportional to $L'(u)$ and therefore linearly dependent. They collapse to $L'(u) = 0$ for the optimal product u^* and impose no constraint on how u^* divides between D and W. A unique (D^*, W^*) cannot be derived from the capacity and data terms under the compute constraint alone; an architecture-sensitive term, such as the thresholded penalties in Equations (4) and (5), is required. We therefore treat those penalties as empirical candidates awaiting validation, not as a proven compute-optimal depth exponent.

A.5 Leave-one-deep-run-out sensitivity

Table 7 lists the per-run leave-one-deep-run-out refits summarized in Section 4.4 and Figure 4. The refitted κ values remain within the full-data bootstrap interval, and the $M_0 - M_3$ LOOCV gap remains positive under each individual deletion.

Table 7: Leave-one-deep-run-out sensitivity. Each row refits the parameter-count (M_0) and LOOCV-preferred log-onset (M_3) models after dropping the deepest run at one large scale. The M_3 advantage is not eliminated by dropping any one of the three deepest large-tier runs individually.

Dropped run	κ	LOOCV M_0	LOOCV M_3	Gap ($M_0 - M_3$)
None (full data)	6.48	0.108	0.102	+0.006
1B/80L	5.06	0.107	0.102	+0.005
3B/72L	6.19	0.108	0.102	+0.006
7B/64L	6.54	0.111	0.104	+0.007

Table 8: Implied onset from the LOOCV-preferred log candidate, $D_{\text{crit}}(W) = \kappa \ln W$ with $\kappa = 6.48$, applied using each run's own width. The complete $W = 512$ sweep remains below the fitted onset; consequently, its small non-monotonicity is not represented by the excess-depth penalty.

Tier/run	D	W	$\kappa \ln W$	Penalty active?
Baseline lowest loss	16	1536	47.5	No
Baseline deepest fixed-W run	32	512	40.4	No
1B_shallow	12	2560	50.9	No
1B_medium	24	1792	48.5	No
1B_deep	48	1280	46.4	Yes
1B_vdeep	64	1152	45.7	Yes
1B_extreme	80	1024	44.9	Yes
3B_shallow	16	3840	53.5	No
3B_medium	24	3072	52.0	No
3B_deep	40	2432	50.5	No
3B_vdeep	56	2048	49.4	Yes
3B_extreme	72	1792	48.5	Yes
7B_wide	32	4096	53.9	No
7B_deep	64	2816	51.5	Yes

A.6 Implied onset depth

Table 8 sets the implied onset $D_{\text{crit}}(W) = \kappa \ln W$ ($\kappa = 6.48$) against the observed depths at each scale, using each run's own width.

B Extended Experimental Details

B.1 Architecture and documented tier-specific training settings

Table 9 distinguishes the model-family settings shared across the grid from the large-scale optimization settings. For the 1B, 3B, and 7B grids, batch size, sequence length, peak learning rate, and warmup are documented and held fixed within tier. Complete matching batch-size and sequence-length metadata are unavailable for the baseline grid.

Rationale for controlled tier-specific protocols. Holding the documented protocol fixed within a scale tier is a deliberate design choice aligned with the question studied here: whether architecture shape matters when its alternatives are trained under the same tier-specific settings. Per-architecture tuning or μP transfer would answer a different question: which shape performs best after its optimization regime is adapted. The absence of such tuning therefore defines the within-tier estimand and limits transfer beyond it; it is not uncontrolled variation within a tier.

Table 9: Shared model-family settings and documented tier-specific large-scale training settings. The 1B, 3B, and 7B values are fixed within their respective architecture grids; they are not shared across tiers.

Hyperparameter	**Value**
Optimizer	AdamW
LR schedule	Warmup cosine decay
Weight decay	0.1
β_1, β_2	Optax AdamW defaults: 0.9, 0.999
Gradient clip (norm)	1.0
Dropout	0.0
Precision	bfloat16
Training seed	42
Tokenizer	GPT-2 (vocabulary size 50,257)
Model family	Decoder-only Pre-LN Transformer with learned absolute positions and GELU
MLP expansion ratio	$d_{\text{ff}} = 4W$
Attention heads	32 in the documented 1B–7B configurations; attention-head dimension therefore varies with W. Baseline head settings are specified in the released configurations
Attention/MLP biases	Enabled (Flax `Dense` defaults)
Embedding tying	Untied input embedding and output projection
LayerNorm	Flax defaults, $\epsilon = 10^{-6}$, scale and bias enabled
Initialization	Flax Dense default (LeCun normal); positional embeddings $\mathcal{N}(0, 0.02)$; no $1/\sqrt{D}$ depth scaling
Documented tier-specific large-scale settings	
1B batch / sequence / peak LR / warmup	524,288 tokens / 2,048 / 3×10^{-4} / 1,000
3B batch / sequence / peak LR / warmup	1,048,576 tokens / 2,048 / 2×10^{-4} / 1,500
7B batch / sequence / peak LR / warmup	2,097,152 tokens / 2,048 / 1.5×10^{-4} / 2,000
Baseline batch / sequence	Not recorded in the released per-run result metadata

B.2 Compute resources

- **Hardware:** baseline, 1B, and 3B runs used Google Cloud TPU v4-32 (on-demand); the two 7B runs used v6e-64 (spot). The 7B runs draw on a different accelerator generation from the smaller tiers, though all losses share the same numerical precision and training implementation; hardware-dependent nondeterminism remains part of the reproducibility boundary.
- **Training time:** roughly four weeks for the campaign end to end.
- **Per-model time:** 13 to 227 hours by size.

C Complete Results

Table 10 lists all 30 runs ordered by parameter count; Table 11 reports the per-scale token and compute ranges.

Table 10: Full result table for all 30 runs, sorted by parameter count.

Model	Scale	D	W	Params (M)	Tokens (B)	Loss
baseline_2L_256W	baseline	2	256	27.4	6.72	4.332
baseline_8L_256W	baseline	8	256	32.2	6.09	4.039
baseline_16L_256W	baseline	16	256	38.5	6.69	3.929
baseline_2L_512W	baseline	2	512	58.0	6.11	3.945
baseline_4L_512W	baseline	4	512	64.3	6.49	3.792
baseline_8L_512W	baseline	8	512	76.9	6.10	3.543
baseline_12L_512W	baseline	12	512	89.5	6.62	3.473
baseline_16L_512W	baseline	16	512	102.1	6.40	3.435
baseline_24L_512W	baseline	24	512	127.3	6.29	3.467
baseline_2L_1024W	baseline	2	1024	128.6	6.43	3.542
baseline_32L_512W	baseline	32	512	152.4	6.40	3.441
baseline_8L_1024W	baseline	8	1024	204.1	6.16	3.406
baseline_2L_1536W	baseline	2	1536	211.8	6.57	3.558
baseline_4L_1536W	baseline	4	1536	268.4	6.34	3.398
baseline_16L_1024W	baseline	16	1024	304.8	6.64	3.128
Model	Scale	D	W	Params (M)	Tokens (B)	Loss
baseline_8L_1536W	baseline	8	1536	381.7	6.15	3.100
baseline_12L_1536W	baseline	12	1536	495.0	6.22	3.067
baseline_16L_1536W	baseline	16	1536	608.3	6.57	3.049
1B_deep	1B	48	1280	1075.2	19.61	2.839
1B_medium	1B	24	1792	1108.8	20.58	2.821
1B_extreme	1B	80	1024	1112.0	20.90	2.978
1B_vdeep	1B	64	1152	1137.7	19.64	2.897
1B_shallow	1B	12	2560	1206.4	20.42	2.847
3B_extreme	3B	72	1792	2958.8	62.09	2.681
3B_vdeep	3B	56	2048	3029.1	57.01	2.585
3B_medium	3B	24	3072	3033.3	62.22	2.534
3B_deep	3B	40	2432	3088.8	60.03	2.519
3B_shallow	3B	16	3840	3225.2	58.98	2.553
7B_deep	7B	64	2816	6379.7	138.63	2.417
7B_wide	7B	32	4096	6863.1	143.06	2.298

Table 11: Per-scale token and estimated compute budgets from run metadata. FLOPs use $6NT$, in units of 10^{21}, and are approximate; the ranges show the budget variation left by the approximate tiers.

Scale	Tokens (B)	Est. FLOPs (10^{21})	Hours
baseline	6.09–6.72	0.001–0.024	12.7–13.8
1B	19.61–20.90	0.127–0.148	37.8–39.6
3B	57.01–62.22	1.036–1.141	114.2–125.8
7B	138.63–143.06	5.307–5.891	217.6–227.4