DEPTHBENCH: MEASURING HOW RESIDUAL CONNECTIONS ENABLE MORE COMPUTATIONAL DEPTH

Anonymous authors
Paper under double-blind review

ABSTRACT

Depth is a natural way to increase the computational capacity in Transformers, yet the contribution of deeper layers can diminish as depth grows larger. Recent approaches enhance normalization (e.g., LayerNorm Scaling) or residual connections (e.g., mHC, AttnRes) to enable better information flow and depth utilization. However, it remains unclear whether they truly translate increased architectural depth into effective computational depth, and whether their reported gains stem from better access to information across depth, or unaccounted-for confounding factors. In this paper, we introduce **DepthBench**, a controlled benchmark for studying computational depth across various architectures. We systematically vary the width–depth aspect ratio ($d_{\text{model}}/n_{\text{layer}}$) from shallow–wide to deep–narrow shapes, while keeping the model size and pre-training recipe fixed. Across 10 representative architectures, we find that the benefit of allocating more capacity to depth is strongly architecture-dependent. Standard Pre-LN and most of its norm- and scaling-based variants provide little benefit and can even degrade performance as models become deeper and narrower, whereas HC and Full AttnRes improve consistently even at extreme deep shapes. These gains extend beyond pre-training loss and consistently translate into improved domain-specific performance and effective computation. Controlled layer-level analyses further show that the gains of HC and Full AttnRes are associated with more effective utilization of additional layers, revealing distinct mechanisms of computational depth across architectures. Overall, our results identify residual connection design as a key determinant of whether depth can serve as a meaningful scaling axis by enabling additional architectural depth to translate into effective computation. Code is submitted.

1 INTRODUCTION

Scaling depth is a fundamental way to increase the computational capacity of Transformers, as more layers allow representations to undergo a longer sequence of nonlinear transformations (Levine et al., 2020; Sanford et al., 2024). However, current LLMs suffer from the curse of depth (Sun et al., 2026), where purely increasing architectural depth does not necessarily yield a proportional increase in model capacity. In conventional Pre-LN Transformers, residual connections make deep networks easier to optimize (Xiong et al., 2020), but they do not ensure that every layer contributes useful computation (Sun et al., 2026). As models become deeper, later-layer updates can become increasingly weak, redundant, or similar to those of preceding layers (Li et al., 2024; Sun et al., 2026; Team et al., 2026b; Liu et al., 2026). Consequently, additional layers may therefore increase nominal depth without comparable gains in effective computation or model quality.

Recent work addresses this problem by modifying normalization or residual propagation. For instance, LayerNorm Scaling (LNS) (Sun et al., 2026) and KEEL (Chen & Wei, 2026) refine normalization, whereas AttnRes (Team et al., 2026b), HC (Zhu et al., 2025), and mHC (Xie et al., 2025) redesign the residual pathway. These methods report improved optimization and model quality at large scales, and related designs are being adopted in frontier LLMs such as Kimi K3 (Team et al., 2026a) and DeepSeek V4 (Xu et al., 2026).

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

However, it remains unclear whether they truly translate increased architectural depth into effective computational depth[1]. Existing methods are typically evaluated with different architectures, training recipes, and system configurations. Their gains therefore cannot be cleanly attributed to improved optimization, better access to information across depth, preservation of diverse features, or differences in capacity and computational overhead.

We therefore still lack a unified and comprehensive understanding of which architectural designs can reliably translate more architectural depth into useful computational depth. This motivates our primary research question:

> *Which architectural designs make additional depth computationally effective?*

Answering this question determines when depth can serve as a reliable scaling axis rather than merely increasing the number of layers.

To address this question, we develop DEPTHBENCH to study when depth pays off under a fixed parameter budget. Varying the aspect ratio ($d_{\text{model}}/n_{\text{layer}}$) reallocates parameters between width and depth, enabling a direct comparison between shallow-wide and deep-narrow architectures with a broad range of aspect ratios for 10 representative residual designs while holding both the parameter count and training recipe fixed. We sweep multiple learning rates for every aspect ratio and report each configuration at its best-performing learning rate, ensuring that differences reflect the architecture rather than a suboptimal learning rate. We then rank model performance across configurations and conduct controlled layer-level analyses to determine whether each approach enables effective aspect ratio scaling by making better use of the greater depth.

Our results reveal several insights about how residual connections affect computational depth:

- **Conventional residual connections do not benefit consistently from deeper architectures at iso-parameter (Section 3).** Pre-LN and its normalization/residual-scaling variants do not benefit from deeper, narrower architectures. Their optima remain at relatively large aspect ratios 42.7–76.0 (e.g., $d_{\text{model}} = 1120$ and $n_{\text{layer}} = 20$), obscuring aspect ratio as a useful scaling axis.
- **New residual designs unleash the power of deeper models even with extremely small aspect ratios (Section 3).** HC and Full AttnRes continue to reduce pre-training loss as models become deeper-narrower under iso-parameter scaling, even at an extreme aspect ratio of 9.1 with $d_{\text{model}} = 640$ and $n_{\text{layer}} = 70$. Effective residual design turns aspect ratio into a practical scaling dimension.
- **Alternative residual mechanisms fundamentally change how computation evolves across depth (Section 4.1, 4.2).** Full AttnRes and HC preserve heterogeneous, layer-specific transformations across layers, unlike the increasingly homogeneous deep-layer representations in Pre-LN.
- **The benefits of depth scaling extend beyond pre-training loss (Section 3 and Section 4.1).** The gains of HC and Full AttnRes transfer to domain-specific evaluations. Layer-wise diagnostics further indicate that these methods make more effective use of the additional computational depth.
- **No free lunch: deep models introduce a systems-level efficiency trade-off (Section 4.4).** Although deeper architectures can improve modeling performance, they increase compute and memory overhead while reducing hardware utilization. Realizing their benefits at scale therefore requires better infrastructure and kernel optimization.

2 PRELIMINARIES & SETUP

2.1 MODERN DEEP TRANSFORMER ARCHITECTURES

Pre-LN Transformers with Residual Connections. Modern LLMs predominantly adopt the Pre-LN Transformer architectures (Xiong et al., 2020), where layer normalization is applied before each transformation branch and the resulting update is added to the residual stream (Baevski & Auli,

[1]Effective computational depth is the amount of sequential computation that a model can exploit through successive layers, as measured by the incremental benefit of additional layers or iterations under a controlled compute budget.

Table 1: Overview of residual update mechanisms for the architectures (Team et al., 2026b). Architectures differ primarily in how layer ℓ receives and aggregates information from preceding layers. *Weight*: whether the mixing coefficients are fixed or input-dependent (dynamic). *Source*: which earlier representations layer ℓ can access.

Method	Update rule	Weight	Source
Baseline			
Pre-LN (Xiong et al., 2020)	$h_\ell = h_{\ell-1} + F_\ell(\mathrm{LN}(h_{\ell-1}))$	Fixed	$h_{\ell-1}$
LayerNorm variants			
Sandwich-LN (Ding et al., 2021)	$h_\ell = h_{\ell-1} + \mathrm{LN}_{\mathrm{out}}(F_\ell(\mathrm{LN}_{\mathrm{in}}(h_{\ell-1})))$	Fixed	$h_{\ell-1}$
LNS (Sun et al., 2026)	$h_\ell = h_{\ell-1} + F_\ell\left(\frac{1}{\sqrt{\ell}}\mathrm{LN}(h_{\ell-1})\right)$	Fixed	$h_{\ell-1}$
DeepNorm (Wang et al., 2024)	$h_\ell = \mathrm{LN}(\alpha h_{\ell-1} + F_\ell(h_{\ell-1}))$	Fixed	$h_{\ell-1}$
KEEL (Chen & Wei, 2026)	$h_\ell = \mathrm{LN}(\alpha h_{\ell-1} + F_\ell(\mathrm{LN}(h_{\ell-1})))$	Fixed	$h_{\ell-1}$
Multi-stream residuals			
HC (Zhu et al., 2025)	$H_\ell = H_{\ell-1}A_\ell + F_\ell(\mathrm{LN}(H_{\ell-1}\alpha_\ell))\,\beta_\ell^\top$	Dynamic	m residual streams
mHC (Xie et al., 2025)	$H_\ell = H_{\ell-1}\mathrm{Sinkhorn}(\widetilde{A}_\ell) + F_\ell(\mathrm{LN}(H_{\ell-1}\sigma(\widetilde{\alpha}_\ell)))\,2\sigma(\widetilde{\beta}_\ell^\top)$	Dynamic	m residual streams
Cross-layer access			
AttnRes (Team et al., 2026b)	*Full:* $h_\ell \propto \sum_{i=0}^{\ell-1}\phi(w_\ell, v_i)v_i, \quad v_0 = h_1,\ v_{i\geq 1} = f_i(h_i)$	Dynamic	$[h_1, \ldots, h_{\ell-1}]$
	Block: $h_\ell \propto \sum_{i=0}^{n-1}\phi(w_\ell, v_i)v_i + \phi(w_\ell, v_n^j)v_n^j$	Dynamic	$[v_0, \ldots, v_{n-1}, v_n^j]$
MoDA (Pre-LN) (Zhu et al., 2026)	$h_\ell = h_{\ell-1} + \mathrm{MoDA}_\ell(\mathrm{LN}(h_{\ell-1}); \mathcal{C}_{<\ell})$	Dynamic	$[h_{\ell-1,\leq t}; h_{0,t}, \ldots, h_{\ell-2,t}]$

2018; Dai et al., 2019). Formally, a Pre-LN sublayer updates the hidden state as:

$$h_\ell = h_{\ell-1} + F_\ell(\mathrm{LN}(h_{\ell-1})) \tag{1}$$

where F_ℓ denotes the attention or feed-forward transformation at layer ℓ, and LN is typically instantiated as RMSNorm (Zhang & Sennrich, 2019). Compared with Post-Layer Normalization (Post-LN) (Ba et al., 2016), Pre-LN substantially improves training stability at large depth by preserving a direct residual pathway for gradient propagation (Xiong et al., 2020; Li et al., 2024).

Pre-LN Issues and Improvements. While Pre-LN largely resolves the *trainability* issue of deep Transformers (Xiong et al., 2020), it does not necessarily guarantee *effective computational depth*: a model can be made very deep without fully utilizing its later layers (Li et al., 2024; Sun et al., 2026). This limitation has been widely described as the **Curse of Depth** (Sun et al., 2026) and **Pre-LN dilution** (Team et al., 2026b), where the contribution of deeper layers progressively weakens.

A growing line of architectures therefore modifies normalization and residual connections, seeking better depth utilization (Chen & Wei, 2026; Sun et al., 2026; Team et al., 2026b; Li et al., 2026). To provide a thorough evaluation, we categorize the architectures considered in this work into four groups, including Pre-LN baseline, normalization variants, multi-stream residuals and cross-layer access, covering 10 representative architectures as summarized in Table 1.

2.2 DepthBench Design

Controlled Width–Depth Aspect Ratio Scaling. Simply increasing depth at fixed width also increases model size and compute, confounding the effect of depth with generic scale. DepthBench instead treats depth as a *capacity allocation* choice, varying $d_{\mathrm{model}}/n_{\mathrm{layer}}$ from shallow–wide to deep–narrow configurations under an approximately fixed parameter budget.

Architectural Backbone. To isolate the effects of norm- and residual- design, we instantiate all variants on a shared LLaMA-like backbone. For sub-1B models, the backbone uses multi-head self-attention (MHA) (Vaswani et al., 2017) with 16 heads, rotary positional embeddings (RoPE) (Su et al., 2024), RMSNorm (Zhang & Sennrich, 2019) with $\epsilon = 10^{-6}$, SwiGLU feed-forward layers (Shazeer, 2020), the GPT-NeoX tokenizer (Black et al., 2022) with a vocabulary size of 50280, and untied input and output embeddings. For the 1.6B models, we follow the Qwen3-1.7B attention configuration (Yang et al., 2025), replacing MHA with grouped-query attention (GQA) (Ainslie et al., 2023) using 16 query heads and 8 key-value heads. All other architectural choices remain unchanged. Architecture-specific implementation details are provided in Appendix B.

Table 2: Overview of the experimental settings in DEPTHBENCH.

Setting	Matched Budget	# Shapes	Aspect Ratios	Purpose
Main benchmark	≈400M total	7	9.1–76.0	Comprehensive architecture comparison
Iso-backbone control	≈300M backbone	6	6.5–78.0	Control for backbone-size variation
Multi-scale suite	200M–500M total	3	≈28, ≈43, ≈76	Test consistency across model scales
Validation at scale	≈1.6B total	3	27.9, 43.2, 73.1	Validate trends at larger scale

Model Configurations. We organize DEPTHBENCH into four complementary experimental suites, summarized in Table 2. Our main benchmark operates at the 400M total size scale and compares 10 architectures across seven model shapes, spanning aspect ratios from 76.0 ($d_{\text{model}} = 1216$, $n_{\text{layer}} = 16$) to 9.1 ($d_{\text{model}} = 640$, $n_{\text{layer}} = 70$). We additionally conduct an *iso-backbone* control, in which the Transformer backbone size, excluding the input embeddings and LM head, is held fixed at 300M parameters. This removes a confound of fixed-total-parameter comparisons: narrower models allocate fewer parameters to the embedding and LM head, and thus a larger fraction of the total budget to the Transformer backbone (Porian et al., 2024). To assess robustness across scale, we evaluate three representative aspect ratios at {200M, 300M, 400M, 500M} parameters. Finally, we test whether the same trends persist at the 1.6B scale under three shapes. Full model configurations for all experimental suites are provided in Appendix C.

For each depth $L = n_{\text{layer}}$, we choose hidden dimension $d = d_{\text{model}}$ such that the relevant parameter budget is approximately matched. For sub-1B models, we set the intermediate dimension to $d_{\text{ff}} = \text{ceil}_{16}\left(8d/3\right)$, where ceil_{16} denotes rounding up to the next multiple of 16. For the 1.6B models, we use $d_{\text{ff}} = 3d$. Since total parameters scale approximately as $N \propto 12Ld^2 + 2Vd$ where V is vocabulary size, increasing depth under a fixed total or backbone parameter budget necessarily requires reducing width. This construction yields a controlled spectrum from shallow–wide to deep–narrow shapes, enabling us to isolate how architectures trade off capacity between width and depth.

2.3 PRE-TRAINING SETTINGS

We use OLMo-core[2] to pre-train all models from scratch on FineWeb-Edu (Penedo et al., 2024), with a token budget of 20 tokens per parameter following the Chinchilla scaling law (Hoffmann et al., 2022) (especially, 8B tokens for 400M models and 32B for 1.6B models), and reserve a disjoint held-out split for evaluation. We use a sequence length of 2048 and a global batch size of 512 sequences, corresponding to approximately 1M tokens per optimization step. Optimization uses AdamW (Loshchilov & Hutter, 2017) with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.1, and gradient clipping at 1.0. We follow the default OLMo-core initialization with normally distributed weights and standard deviation 0.02. The learning rate follows cosine decay with a 10% linear warmup and decays to 10% of its peak value (Loshchilov & Hutter, 2016).

For 400M models, we perform architecture-specific learning rate sweeps over $\{5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}, 5 \times 10^{-3}\}$, except for LNS, for which we use $\{2 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-2}, 2 \times 10^{-2}\}$. The optimal learning rate is consistent across width–depth shapes within each architecture, yielding 1×10^{-2} for LNS, 1×10^{-3} for DeepNorm, and 2×10^{-3} for all others. We reuse these learning rates for the iso-backbone and multi-scale experiments, and use 5×10^{-4} for 1.6B models following common practice (Sun et al., 2026; Li et al., 2024). All other pre-training settings are held fixed across architectural variants and aspect ratios.

3 MAIN RESULTS

Figure 1 shows validation loss on the 400M benchmark across a wide range of architectures and aspect ratios. Figure 2 shows validation loss across aspect ratios of Pre-LN, Full AttnRes and HC in 200M–500M and 1.6B suites. To complement the validation loss, we further evaluate teacher-forced negative log-likelihood (NLL) on coding, STEM, and math tasks in Figure 3, following the NLL-based pre-training evaluation protocol in (Team, 2026; Chen et al., 2026). We observe a clear pattern: whether increasing depth helps strongly depends on the residual connections.

[2]https://github.com/allenai/OLMo-core

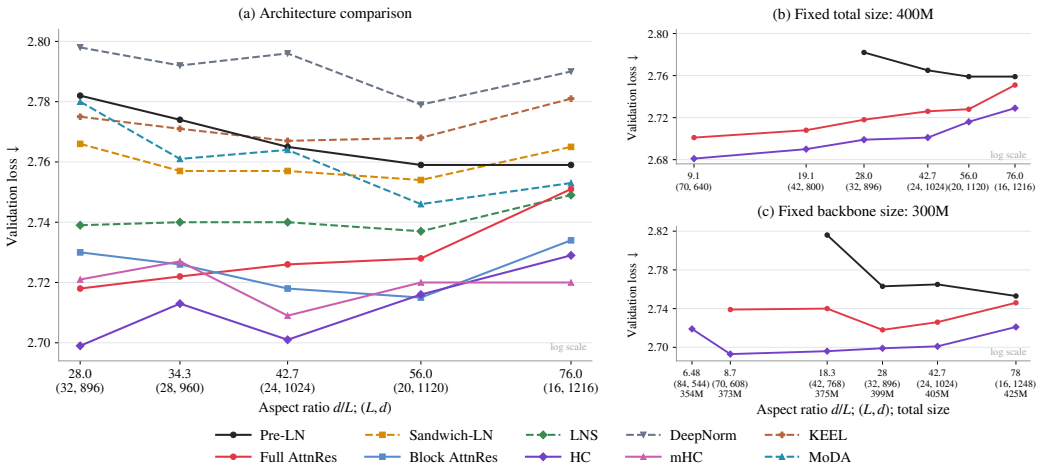

Figure 1: **Left:** 400M main results of validation loss across 10 architectures. **Right:** Validation loss across a wider range of shapes, including extreme small aspect ratios, under fixed total size (400M, top) and fixed backbone size (300M, bottom) for Pre-LN, HC and Full AttnRes.

Pre-LN and normalization variants are largely insensitive or even unfavorable to increasingly deep–narrow shape. As shown in Figure 1 (a), for Pre-LN, validation loss monotonically increases from 2.759 at $L = 16$ to 2.782 at $L = 32$, showing that reallocating parameters from width to depth hurts performance. Sandwich-LN, LNS, DeepNorm, KEEL and MoDA also show either weak or non-monotonic trends, with their optima occurring at intermediate or shallower shapes.

HC and Full AttnRes scale favorably with depth and even surprisingly continue to improve at extremely deep shapes. In contrast, both HC and Full AttnRes consistently improve as models become deeper and narrower. Full AttnRes steadily improves from 2.751 at $L = 16$ to 2.718 at $L = 32$, while HC improves from 2.729 at $L = 16$ to 2.699 at $L = 32$. This trend is the opposite of Pre-LN, suggesting that these new residual designs strongly prefer deep shapes.

We further extend the scaling range, up to 70 layers. As shown in Figure 1 (b-c), the favorable trend persists in both fixing total size and backbone size. Notably, with the backbone size fixed, deep–narrow models improve even as the total size decreases. This suggests that their depth-scaling gains arise from the width–depth allocation itself rather than simply from increased backbone size. Overall, HC and Full AttnRes are exceptionally well suited to depth scaling, with performance continuing to improve as models are pushed toward increasingly deep–narrow, even extreme, shapes.

The favorable width-depth scaling behavior of HC and Full AttnRes is less evident in their derived variants, Block AttnRes and mHC. As shown in Figure 1 (a), Block AttnRes remains competitive but performs best at an intermediate shape, reaching 2.715 at $L = 20$ before degrading to 2.730 at $L = 32$. Similarly, mHC achieves strong overall performance and its best performance at $L = 24$ but shows no consistent gain with increasing depth.

A wide range of scales coupled with domain-specific evaluation shows the consistent trend. As shown in Figure 2, across 200M–500M models, Full AttnRes consistently benefits from deeper–narrower shapes, and this trend persists at 1.6B, suggesting that its favorable depth scaling extends beyond the 400M regime. HC shows a similar trend at smaller scales, but exhibits less stable behavior as scale increases: the 500M run at the largest aspect ratio encounters gradient explosion, while the 1.6B results do not show the same clear improvement with depth. The reason is that HC is more sensitive to optimization hyperparameters and may require finer learning-rate tuning across scales, consistent with the motivation of mHC to improve the large-scale optimization stability of unconstrained HC (Zhu et al., 2025; Xie et al., 2025).

As shown in Figure 3, deeper HC and Full AttnRes models generally achieve lower NLL across coding, STEM, and math evaluations. The trend is particularly clear for STEM and math, where

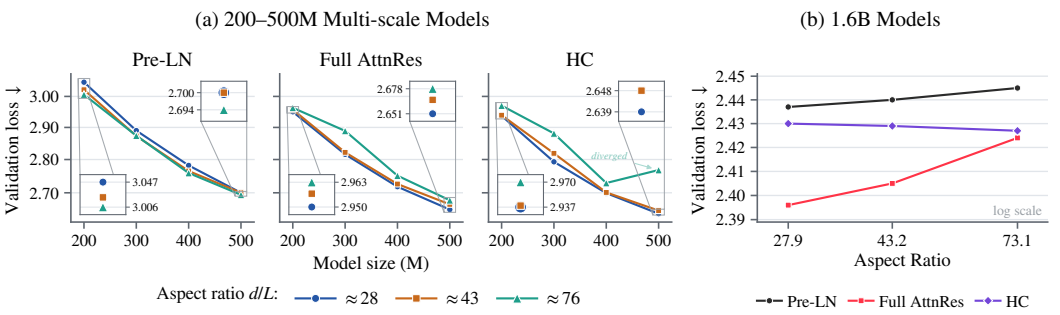

Figure 2: Validation loss for Pre-LN, Full AttnRes, and HC under three aspect ratios. **Left:** {200M, 300M, 400M, 500M} models. **Right:** 1.6B models.

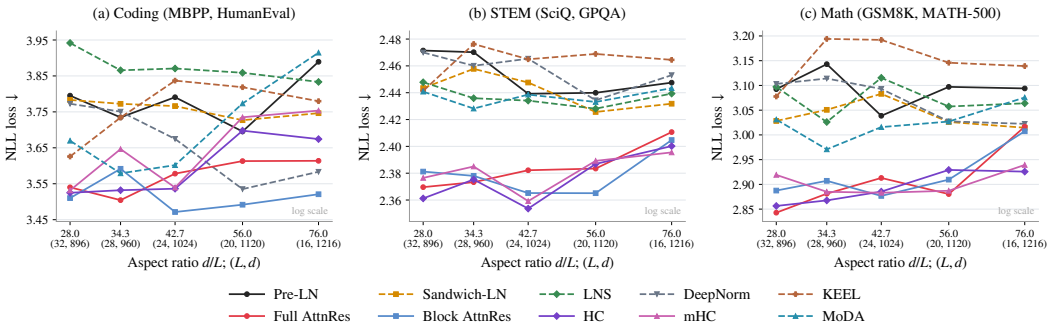

Figure 3: NLL results of 400M models on Coding, STEM and Math categories. We report negative log-likelihood across model aspect ratios and architectures. Each panel shows the average NLL over two representative tasks: MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021) for **coding (left)**, SciQ (Welbl et al., 2017) and GPQA (Rein et al., 2023) for **STEM (middle)**, and GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2024) for **math (right)**.

both architectures consistently improve as capacity is shifted toward depth. In contrast, Pre-LN and most other variants show weaker or non-monotonic trends. These results suggest that the favorable width-depth scaling translates to domain-specific predictive capability rather than only lower held-out pre-training loss.

Overall, our results identify HC and Full AttnRes as the two architectures with the clearest favorable depth-scaling behavior. We next study how these architectures utilize their layers.

4 Analysis

4.1 Evaluating Depth Utilization Across Architectures

Prior work has shown that deep layers in Pre-LN Transformers can become increasingly ineffective (Sun et al., 2026). We therefore ask whether the favorable aspect ratio scaling of those new residual designs is accompanied by more effective utilization across their layers.

We study whether each layer is functionally important by adopting *causal scores* and *permutation scores* from (Muhtar et al., 2026; Csordás et al., 2025) to quantify how strongly individual layers [3] affect subsequent computation and how interchangeable different layers are. For a skipped layer s and a subsequent layer $\ell > s$, the causal score is defined as:

$$C(s,\ell) = \frac{\left\|(h_{\ell+1} - h_\ell) - (\bar{h}_{\ell+1} - \bar{h}_\ell)\right\|_2}{\|h_{\ell+1} - h_\ell\|_2}, \tag{2}$$

[3] A layer contains attention and feed-forward sub-layers, bounded by h_ℓ or H_ℓ as in Table 1.

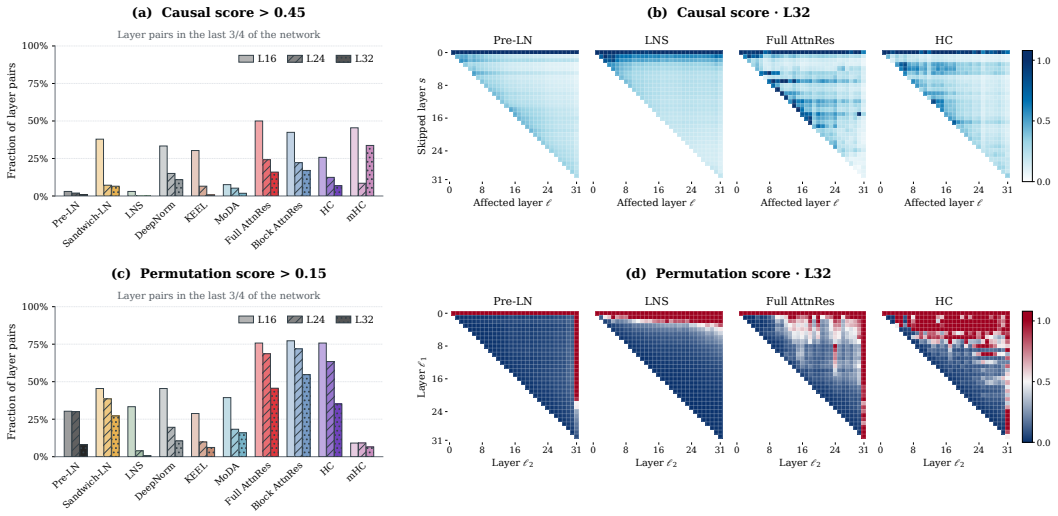

Figure 4: **Left:** Fraction of layer pairs in the last $3/4$ of the network exceeding the causal-score (> 0.45) and permutation-score (> 0.15) thresholds, respectively. **Right:** Pairwise causal and permutation score.

where h denotes the original hidden states and $\bar{h}$ denotes the hidden states obtained after skipping layer s. A larger $C(s, \ell)$ means that removing layer s more strongly changes the update performed by a subsequent layer ℓ, indicating stronger causal dependence across depth.

We additionally measure how sensitive the model is to exchanging the order of two layers. For layers ℓ_1 and ℓ_2, the permutation score is defined as:

$$P(\ell_1, \ell_2) = \frac{\left|\mathcal{L}(M) - \mathcal{L}(M_{\text{swap}(\ell_1,\ell_2)})\right|}{\mathcal{L}(M)}, \tag{3}$$

where $\mathcal{L}(M)$ denotes the language modeling loss of the original model and $\mathcal{L}(M_{\text{swap}(\ell_1,\ell_2)})$ denotes the loss of the model after swapping layers ℓ_1 and ℓ_2. A larger permutation score indicates that exchanging the two layers causes a larger performance degradation, suggesting that they perform more specialized and order-dependent computations.

To avoid the universally important early layers dominating the statistics, we restrict attention to pairs whose first layer lies in the last three quarters of the network. We report R_{causal}, the fraction of pairs (s, ℓ) with $s \geq \lceil L/4 \rceil$ and $\ell > s$ for which $C(s, \ell) > 0.45$, and R_{perm}, defined analogously for pairs (ℓ_1, ℓ_2) satisfying $P(\ell_1, \ell_2) > 0.15$.

As shown in Figure 4, Pre-LN, normalization variants, and MoDA exhibit few strong late-layer interactions. Only 1% of Pre-LN causal scores exceed 0.45, compared with 16% for Full AttnRes and 7% for HC. For permutation scores, the corresponding fractions of Pre-LN, Full AttnRes and HC are 8%, 46%, and 35%. DeepNorm and KEEL remain low despite the last-three-quarters restriction, which could favor Post-LN-based methods. mHC likewise shows a marked drop in permutation sensitivity, consistent with its weaker width-depth scaling. The pairwise heatmaps show the same pattern, with substantially stronger cross-layer dependence and order sensitivity in Full AttnRes and HC. Complete results are provided in Appendix G.

Overall, Pre-LN, most norm- variants, and MoDA exhibit weak and partly interchangeable late-layer computation, whereas Full AttnRes and HC retain stronger layer-specific causal dependence and order sensitivity, suggesting that their additional depth remains more functionally consequential.

4.2 Characterizing Depth Processing in Full AttnRes and HC

Section 4.1 shows that AttnRes and HC retain stronger cross-layer dependence and order sensitivity than Pre-LN and its normalization variants. We next examine whether these architectures realize more effective depth through different computational mechanism.

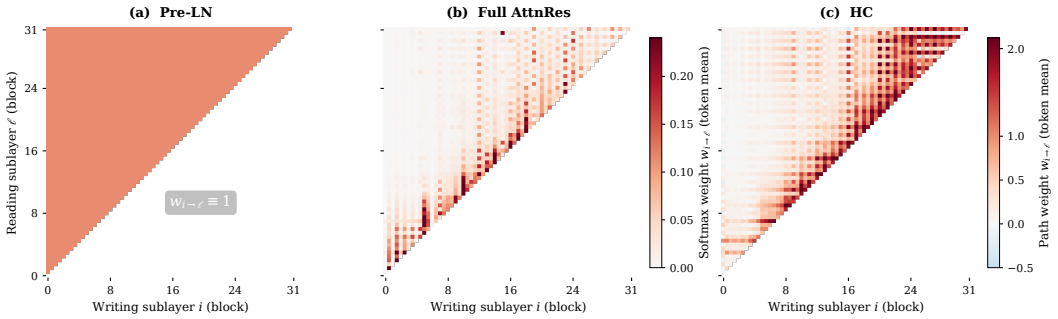

Figure 5: **Explicit residual-path weights $w_{i\to\ell}$.** Each cell shows the mixing weight of sublayer output f_i to the pre-norm input of a later sublayer ℓ, averaged over validation calibration tokens. **(a) Pre-LN:** $w_{i\to\ell} \equiv 1$. **(b) Full AttnRes:** $w_{i\to\ell}$ is the per-token softmax weight on f_i. **(c) HC:** $w_{i\to\ell} = \beta_i^\top A_{i+1} \cdots A_{\ell-1}\alpha_\ell$, combining write mappings, residual mappings, and read mappings.

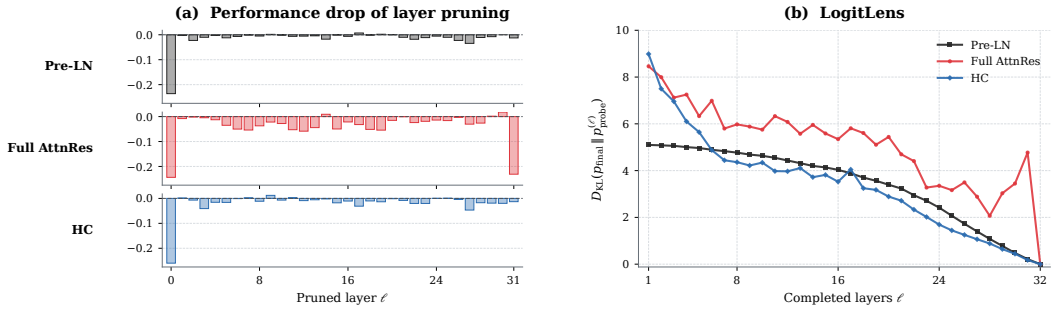

Figure 6: **Left:** Performance drop after removing a single layer. **Right:** The KL divergence between layerwise and final prediction distributions.

Explicit paths across depth. Figure 5 traces how earlier sublayer outputs contribute to later computations. Pre-LN accumulates all updates in a single residual stream, giving $w_{i\to\ell} \equiv 1$. Full AttnRes instead forms a normalized, non-negative mixture over the embedding and all preceding sublayer outputs, providing direct access to earlier computations. HC realizes cross-depth access recursively, with the contribution of branch output f_i to branch ℓ given by $w_{i\to\ell} = \beta_i^\top A_{i+1} \cdots A_{\ell-1}\alpha_\ell$. Its learned paths are strongest locally but remain substantial over longer ranges. Thus, Full AttnRes directly retrieves stored earlier outputs, whereas HC propagates and recombines them through interacting residual streams.

Functional consequences. We first prune individual layers and measure the resulting change in ARC-Easy accuracy (Figure 6 (a)). Pre-LN and HC show similar patterns: both are dominated by the first layer, while pruning most later layers causes only modest degradation. Full AttnRes is sensitive across a broader range, with strong dependence on both its first and final layers.

We further apply LogitLens (Csordás et al., 2025) to track how predictions evolve across depth. For Pre-LN, we decode the current residual stream; for Full AttnRes, the mixture produced by the next layer's depth-mixing query; and for HC[4], the sum of its four residual streams. For probe state h_ℓ, we compute $p^{(\ell)}_{\text{probe}} = \text{softmax}(W_{\text{LM}}\,\text{Norm}(h_\ell))$ and report $D_{\text{KL}}(p_{\text{final}} \| p^{(\ell)}_{\text{probe}})$. As shown in Figure 6 (b), Pre-LN approaches the final prediction monotonically, consistent with progressive refinement of a single residual stream. HC follows a similarly smooth trajectory, but its earlier representations remain farther from the final prediction. Full AttnRes is markedly less monotonic, consistent with continued retrieval and late integration of stored sub-layer output sources.

[4]We implement HC following the original HC paper (Zhu et al., 2025), where the final output is an sum of the four residual streams. Later HC/mHC implementations, such as DeepSeek V4 (Xu et al., 2026), may instead use a learnable final mixer.

Together, these results suggest that Full AttnRes and HC need not make every layer individually indispensable. Rather, their learned cross-depth routing preserves differentiated access to earlier computations, maintaining stronger cross-layer dependence and order sensitivity instead of reducing later layers to largely interchangeable refinements.

4.3 Potential Hidden Cost for mHC and Block AttnRes

The scaling behavior of Full AttnRes and HC is less evident in their derived variants, Block AttnRes and mHC. Their weaker depth scaling appears associated with architectural constraints: fixed block boundaries in Block AttnRes reduce effective transformations at early layers, while mHC produces increasingly aligned, lower-rank residual transport. We defer detailed analysis to Appendix D.

4.4 No Free Lunch: The Accuracy–Efficiency Trade-off of Depth Scaling

Depth scaling is not free even under a fixed parameter budget. Since $N \approx 12Ld^2 + 2Vd$, while the dominant projection cost proportional to Ld^2 remains approximately constant, quantities proportional to Ld increase when scaling depth. For example, during prefill with sequence length T, attention prefill cost scales as $\mathcal{O}(T^2Ld)$ and becomes increasingly important in the long-context regime $T \gg d$, while KV cache memory similarly scales as $\mathcal{O}(TLd)$. We provide the FLOPs counting analysis in Appendix E. Moreover, the practical latency overhead can exceed the increase suggested by FLOPs alone, as deeper models incur lower hardware utilization. We also show in Appendix F an example of the empirical prefill cost during inference, with respect to FLOPs, KV cache cost, peak memory, and GPU hours. Especially, in our setting, with prefill sequence length 4096 on an NVIDIA A100-SXM4-40GB GPU, Full AttnRes at the 70-layer shape ($L = 70$, $d = 640$) requires approximately 8.8 GPU-hours per 1B tokens, compared with 4.5 GPU-hours for Pre-LN at the same shape and approximately 4.0 GPU-hours for a shallower 32-layer Full AttnRes model ($L = 32$, $d = 896$). Notably, those algorithmic and empirical forward costs do not directly reflect training wall clock. Deeper models incur more sequential execution and smaller matrix multiplications, so their practical cost depends strongly on hardware, kernels, parallelism, and memory behavior. Realizing the gains of depth scaling therefore requires architecture–system co-design.

5 Related Work

Below we provide a brief sketch of related work. A more complete review is deferred to Appendix A.

Scaling-law and model aspect ratio studies examine how width–depth allocation affects model performance (Kaplan et al., 2020; Hoffmann et al., 2022; Tay et al., 2021; Petty et al., 2024; Liu et al., 2024; McLeish et al., 2025). Meanwhile, work on deep Transformers has progressed from improving trainability through normalization and residual scaling (Xiong et al., 2020; Wang et al., 2024; Ding et al., 2021) to improving effective layer utilization (Li et al., 2024; Sun et al., 2026; Chen & Wei, 2026; Li et al., 2026; Trochelmann et al., 2026). More recent architectures further replace fixed residual accumulation with learnable routing, including HC/mHC (Zhu et al., 2025; Xie et al., 2025) and AttnRes (Team et al., 2026b). Our work compare those designs under controlled aspect ratio scaling to study when architectural depth translates into effective computation.

6 Conclusion

In this work, we introduce DepthBench, a controlled benchmark for studying when architectural depth translates into effective computational depth in Transformers. By systematically varying width–depth aspect ratios under approximately fixed parameter and training budgets, we show that the benefit of allocating more capacity to depth is strongly architecture-dependent. In particular, HC and Full AttnRes consistently benefit from deeper and narrower model shapes, while Pre-LN and most normalization-based variants show much weaker or even unfavorable depth scaling. Our layer-wise analyses further suggest that these gains are associated with more heterogeneous, consequential, and order-sensitive computation across depth, rather than simply improved trainability. Finally, these gains come with additional efficiency costs, highlighting an important performance–efficiency trade-off. Overall, our results identify residual connection design as a key determinant of whether depth can serve as a meaningful scaling axis for Transformer models.

7 AI USE STATEMENT

In this work, we used generative AI tools for reviewing and polishing codebase, formatting tables and figures, improving the clarity and readability of the text. We have not used generative AI tools for research ideation, experimental design, data generation, formulating and proving claims. We have reviewed all AI-assisted work, and take responsibility for the final content of this work, including text and code produced with the aid of generative AI.

8 ETHICS STATEMENT

In this work, we carefully ensure that all experimental protocols conform to established ethical guidelines. Our investigation centers on residual connections as a key component to improve effective computational depth in LLMs, contributing to more sustainable AI practices. Every code framework and dataset employed in this research is obtained from openly accessible sources, guaranteeing respect for intellectual property and protection of personal privacy.

9 REPRODUCIBILITY STATEMENT

We made several efforts to ensure reproducibility. First, we provide architecture implementation details in Appendix B and all pre-training settings in Section 2.3. Second, we controlled for stochasticity by running experiments with fixed random seed 42. Finally, the code and detailed instructions are included in the supplementary materials to facilitate full reproduction of our results.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4895–4901, 2023.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*, 2018.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. Gpt-neox-20b: An open-source autoregressive language model, 2022. URL https://arxiv.org/abs/2204.06745.

Chen Chen and Lai Wei. Post-layernorm is back: Stable, expressive, and deep. *arXiv preprint arXiv:2601.19895*, 2026.

Huanran Chen, Huaqing Zhang, Xiao Li, Yinpeng Dong, Ke Shen, and Jun Zhu. Nexus: Same pretraining loss, better downstream generalization via common minima. *arXiv preprint arXiv:2604.09258*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Róbert Csordás, Christopher D Manning, and Chris Potts. Do language models use their depth efficiently? In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 160313–160362. Curran Associates, Inc., 2025. doi: 10.52202/085713-5357. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/eae3af0f5868f0a2eceb74208966d55b-Paper-Conference.pdf.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 2978–2988, 2019.

Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. Cogview: Mastering text-to-image generation via transformers. *Advances in neural information processing systems*, 34:19822–19835, 2021.

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.

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.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 2261–2269. Ieee, 2017.

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.

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

Yoav Levine, Noam Wies, Or Sharir, Hofit Bata, and Amnon Shashua. Limits to depth efficiencies of self-attention. *Advances in Neural Information Processing Systems*, 33:22640–22651, 2020.

Pengxiang Li, Lu Yin, and Shiwei Liu. Mix-ln: Unleashing the power of deeper layers by combining pre-ln and post-ln. *arXiv preprint arXiv:2412.13795*, 2024.

Tianyu Li, Dongchen Han, Zixuan Cao, Haofeng Huang, Mengyu Zhou, Ming Chen, Erchao Zhao, Xiaoxi Jiang, Guanjun Jiang, and Gao Huang. Siamesenorm: Breaking the barrier to reconciling pre/post-norm. *arXiv preprint arXiv:2602.08064*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Yizhou Liu, Sara Kangaslahti, Ziming Liu, and Jeff Gore. Inverse depth scaling from most layers being similar. *arXiv preprint arXiv:2602.05970*, 2026.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *arXiv preprint arXiv:2402.14905*, 2024.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

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

Sean McLeish, John Kirchenbauer, David Miller, Siddharth Singh, Abhinav Bhatele, Micah Goldblum, Ashwinee Panda, and Tom Goldstein. Gemstones: A model suite for multi-faceted scaling laws. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 123419–123454. Curran Associates, Inc., 2025. doi: 10.52202/085713-4114. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b2b781badeeb49896c4b324c466ec442-Paper-Conference.pdf.

Gaurav Menghani, Ravi Kumar, and Sanjiv Kumar. Laurel: Learned augmented residual layer. *arXiv preprint arXiv:2411.07501*, 2024.

Dilxat Muhtar, Xinyuan Song, Sebastian Pokutta, Max Zimmer, Nico Pelleriti, Thomas Hofmann, and Shiwei Liu. When does sparsity mitigate the curse of depth in llms. *arXiv preprint arXiv:2603.15389*, 2026.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *Advances in neural information processing systems*, 37:136479–136508, 2024.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

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.

Tomer Porian, Mitchell Wortsman, Jenia Jitsev, Ludwig Schmidt, and Yair Carmon. Resolving discrepancies in compute-optimal scaling of language models. *Advances in Neural Information Processing Systems*, 37:100535–100570, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Transformers, parallel computation, and logarithmic depth. *arXiv preprint arXiv:2402.09268*, 2024.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models, 2026. URL https://arxiv.org/abs/2502.05795.

Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*, 2021.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026a.

Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026b.

The Microsoft AI Team. Mai-thinking-1: Building a hill-climbing machine. Technical report, Microsoft AI, 2026. URL https://microsoft.ai/pdf/mai-thinking-1.pdf.

Leon A Trochelmann, Sajad Movahedi, Shiwei Liu, and Antonio Orvieto. Kitenorm: Variance regularisation for stable and scalable post-ln transformers. In *High-dimensional Learning Dynamics 2026*, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In Leon Derczynski, Wei Xu, Alan Ritter, and Tim Baldwin (eds.), *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4413. URL https://aclanthology.org/W17-4413/.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, et al. mhc: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

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

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Boyi Zeng, Yiqin Hao, Zitong Wang, Shixiang Song, He Li, Feichen Song, Yifan Liu, Ziwei He, Xinbing Wang, and Zhouhan Lin. Depth-attention: Cross-layer value mixing for language models. *arXiv preprint arXiv:2606.05014*, 2026.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations*, volume 2025, pp. 97183–97219, 2025.

Lianghui Zhu, Yuxin Fang, Bencheng Liao, Shijie Wang, Tianheng Cheng, Zilong Huang, Chen Chen, Lai Wei, Yutao Zeng, Ya Wang, et al. Mixture-of-depths attention. *arXiv preprint arXiv:2603.15619*, 2026.

A Related Work

Scaling Laws and Model Aspect Ratios. Scaling laws have primarily characterized language modeling performance as a function of model size, data, and compute (Kaplan et al., 2020; Hoffmann et al., 2022). Early studies typically find that width and depth have comparatively limited effects once model size was controlled (Kaplan et al., 2020). However, subsequent work has shown that how capacity is allocated across architectural shapes can matter. Levine et al. (2020) characterizes distinct depth-efficient and depth-inefficient regimes in self-attention. Tay et al. (2021) and Petty et al. (2024) show that model shape can substantially affect downstream fine-tuning performance, implying that deep models potentially have better generalization. Relatedly, MobileLLM (Liu et al., 2024) demonstrates the practical effectiveness of deep-and-thin architectures in the sub-billion-parameter regime. More recently, Gemstones (McLeish et al., 2025) systematically studies scaling across diverse model shapes and shows that scaling-law prescriptions can be sensitive to width–depth choices and experimental design.

These studies, however, largely examine model shape on fixed conventional model architectures. In contrast, we study width–depth allocation across different normalization and residual architectures and show that whether reallocating capacity from width to depth is beneficial depends strongly on residual connection design.

Deep Transformer Training and the Curse of Depth. The original Post-LN Transformer (Ba et al., 2016; Vaswani et al., 2017) can exhibit unstable gradients, while Pre-LN substantially improves optimization stability by providing better gradient propagation (Xiong et al., 2020). Subsequent work further improves the trainability of deep Transformers through normalization, residual scaling, and initialization. For example, DeepNorm (Wang et al., 2024) combines residual scaling with depth-aware initialization to stably train Post-LN Transformers with up to thousands of layers, while Sandwich-LN (Ding et al., 2021; Kim et al., 2025) improves activation-variance control and gradient stability through putting normalization both before and after the transformation branch.

More recent work has shifted attention from merely training deep models to whether their additional layers are effectively utilized. Mix-LN (Li et al., 2024) shows that Pre-LN can produce increasingly weak gradients in deeper layers and combines Pre- and Post-LN to improve their contribution. Similarly, the Curse of Depth (Sun et al., 2026) attributes ineffective deep layers in Pre-LN Transformers to depth-dependent residual-stream variance growth and introduces LayerNorm Scaling to strengthen deep-layer updates. Recent approaches further revisit the Pre-/Post-Norm trade-off: KEEL (Chen & Wei, 2026) stabilizes very deep Post-LN Transformers through highway-style residual connections, SiameseNorm (Li et al., 2026) decouples Pre- and Post-Norm behavior using two coupled streams, and KiteNorm (Trochelmann et al., 2026) regularizes hidden-state variance to stabilize Post-LN training.

These studies establish that trainability and depth utilization are distinct: a Transformer can be optimized successfully at large architectural depth while still deriving limited benefit from its later layers. Our work takes the next step by asking whether improved deep-layer utilization is sufficient to make depth a favorable scaling axis under a fixed parameter budget.

Residual Connections. Residual connections are a fundamental mechanism for enabling information and gradient propagation through deep networks. Highway Networks (Srivastava et al., 2015) introduced gated shortcuts to regulate information flow across layers, while ResNet (He et al., 2016) established identity residual connections as a simple and effective means of optimizing substantially deeper networks. Beyond local identity shortcuts, DenseNet (Huang et al., 2017) showed the benefit of providing layers with direct access to earlier representations, motivating richer forms of cross-depth information reuse.

Recent work has generalized this idea in Transformers by making residual propagation increasingly learnable. DenseFormer (Pagliardini et al., 2024) mixes current and preceding representations through learned depth-weighted averaging, while LAuReL (Menghani et al., 2024) augments the canonical residual layer with learnable residual mappings. Hyper-Connections (HC) (Zhu et al., 2025) expand the single residual stream into multiple streams with learned read, write, and mixing operations. mHC (Xie et al., 2025) further constrains these mappings to restore the identity-preserving properties required for stable large-scale training. Complementarily, Attention Residuals

(AttnRes) (Team et al., 2026b) replace fixed residual accumulation with input-dependent attention over preceding sub-layer outputs, enabling selective access to computations across depth. Related approaches such as MoDA (Zhu et al., 2026) and Depth-Attention (Zeng et al., 2026) provide direct cross-layer access through depth-wise key–value cache representations.

While these methods demonstrate that richer residual pathways can improve optimization and model quality, they are typically evaluated under different model shapes and training settings. We instead study them under controlled width–depth aspect ratio scaling, asking whether their residual topology actually converts additional architectural depth into effective computation.

B Architecture-specific Settings

All variants in Section 2.1 share the LLaMA-like backbone. Here we record every architecture-specific choice that deviates from it. Table 3 summarises the resulting parameter counts.

Table 3: Total parameters and overhead relative to Pre-LN at $L = 24$ (d_{model}=1024).

Architecture	Parameters	Δ vs. Pre-LN
Pre-LN / LNS / DeepNorm	405.41M	—
Sandwich-LN	405.46M	+0.049M
KEEL	405.45M	+0.048M
AttnRes (Full / Block)	405.51M	+0.098M
HC (n=4)	405.70M	+0.296M
mHC (n=4)	410.13M	+4.72M
MoDA	453.64M	+48.2M

Pre-LN (Xiong et al., 2020). The reference block $h_\ell = h_{\ell-1} + F_\ell(\mathrm{RMSNorm}(h_{\ell-1}))$ with one pre-normalisation per sub-layer and no additional parameters.

Sandwich-LN (Ding et al., 2021). Each sub-layer output is normalised a second time before the residual addition, $h_\ell = h_{\ell-1} + \mathrm{RMSNorm}_{\text{out}}\big(F_\ell(\mathrm{RMSNorm}_{\text{in}}(h_{\ell-1}))\big)$, giving four RMSNorms per block. The output norms are initialised to the identity (unit scale). This adds $2d$ parameters per block.

LayerNorm Scaling (LNS) (Sun et al., 2026). The normalised input of both sub-layers in block ℓ (1-indexed) is multiplied by $1/\sqrt{\ell}$ before entering attention and the feed-forward network, and the same factor is used for both sub-layers of a block, so it ranges from 1 in the first block to $1/\sqrt{L}$ in the last (e.g., 0.20 at L=24, 0.18 at L=32). LNS introduces no parameters but requires a 5× larger optimal learning rate (1×10^{-2}) than the other variants.

DeepNorm (Wang et al., 2024). Based on Post-LN (Ba et al., 2016), DeepNorm up-scales the residual connection before performing layer normalization.

$$h_\ell = \mathrm{LN}\big(\alpha\, h_{\ell-1} + F_\ell(h_{\ell-1})\big), \qquad \alpha = (2L)^{1/4},$$

with α ranging from 2.38 at L=16 to 2.83 at L=32. Following the original recipe (Wang et al., 2024), the block normalisation is a mean-centring LayerNorm (without bias) rather than RMSNorm, while the head normalisation remains RMSNorm. Initialisation follows DeepNet (Wang et al., 2024): the value and output projections of attention and all three feed-forward matrices are scaled by $\beta = (8L)^{-1/4}$ (e.g., std 0.02, $\beta \approx 0.0054$ at L=24); query, key, embedding, and head weights keep std 0.02. DeepNorm is trained with optimal learning rate 1×10^{-3}.

KEEL (Chen & Wei, 2026). KEEL combines pre- and post-normalisation with a large residual gain,

$$h_\ell = \mathrm{RMSNorm}_{\text{post}}\big(\alpha\, h_{\ell-1} + F_\ell(\mathrm{RMSNorm}_{\text{pre}}(h_{\ell-1}))\big), \qquad \alpha = \#\texttt{sub-layers},$$

where α counts attention and feed-forward sub-layers separately. The first block is treated specially: its attention sub-layer is a plain Pre-LN update (no post-norm, no residual up-scaling), and its feed-forward sub-layer is both pre-normalized and post-normalised without residual up-scaling. KEEL adds d parameters in the first block and $2d$ in every other block.

AttnRes (Full) (Team et al., 2026b). Every sub-layer replaces the residual sum by a softmax-weighted mixture over the outputs of all preceding sub-layers and the embedding, i.e. the sources are individual sub-layer outputs rather than accumulated hidden states. The attention sub-layer of block ℓ mixes $2\ell{+}1$ sources, the feed-forward sub-layer $2\ell{+}2$, and the output head $2L{+}1$ (e.g., 49 at $L{=}24$). Each sub-layer owns a learned query vector $\mathbf{w} \in \mathbb{R}^d$ and a per-query RMSNorm gain applied to the keys, and scores are $\mathbf{w}^\top \mathtt{rmsnorm}(\mathbf{v_i})$ with softmax temperature 1 (no $1/\sqrt{d}$ scaling). The mixture is RMS-normalised with the consuming sub-layer's pre-norm weights inside the fused kernel, so the standard pre-normalisation is folded into the mixing step, and the Language modeling (LM) head likewise consumes a mixture normalised with the LM head norm. All query vectors, including the LM head query, are zero-initialised, so training starts from uniform averaging over sources. The first block's attention reads the embedding directly and has no query. Query vectors and key gains are subject to the global weight decay. AttnRes adds $2d$ parameters per sub-layer plus $2d$ for the head.

AttnRes (Block) (Team et al., 2026b). Block AttnRes uses the same block, kernel, initialisation, and parameter count as Full AttnRes and differs only in the depth mixing schedule. Sub-layer outputs are accumulated into a running sum like Pre-LN. At every b-th sub-layer (block size $b = (2L)/8$: $4, 5, 6, 7, 8$ sub-layers for $L = 16, 20, 24, 28, 32$) the running sum is frozen as a new source and reset. Each mixture is taken over the frozen block sums and the current running sum, so the network is always partitioned into $2L/b = 8$ blocks[5] and the head mixes the embedding with eight block outputs. For even b the boundaries fall on attention sub-layers only; for odd b ($L{=}20$, 28) they alternate between attention and feed-forward sub-layers.

Hyper-Connections (HC) (Zhu et al., 2025). We use $n = 4$ residual streams, with an HC connector replacing the residual addition around each pre-normalised attention and FFN branch. Routing combines learned static coefficients with input-dependent $\tanh$ corrections projected from each affine-free RMS-normalised stream. Routing parameters use BF16. Dynamic projections are zero-initialised, with two learned scalar gains initialised to 0.01. The initial residual map is the identity. Zero-indexed sub-layer k reads only stream $k \bmod n$ and writes its output to all streams, with unit read and write weights. Attention output and FFN down-projections are additionally scaled by $1/\sqrt{n} = 1/2$ at initialisation. Static routing weights receive no weight decay, while dynamic projections use 0.1. Embeddings are replicated across streams, which are summed before the final RMSNorm and LM head.

Manifold-Constrained Hyper-Connections (mHC) (Xie et al., 2025). We use Liger Kernel[6] with $n = 4$ streams and 20 Sinkhorn–Knopp iterations. Each attention and FFN connector replaces the standard residual addition around the pre-normalised branch. A linear projection of the affine-free RMS-normalised, concatenated streams produces input-dependent routing logits: read weights use sigmoid, write weights use twice sigmoid, and residual maps use Sinkhorn normalisation, without $\tanh$. RMS and Sinkhorn epsilons are 10^{-6}. The read-weight epsilon is zero. Projection weights and branch computation use BF16, while routing biases, learned gains, and coefficient normalisation use FP32.

We initialise the dynamic projection Φ to zero and its three scalar gains to 0.01. At zero-indexed sub-layer k, read biases are $+8$ for stream $k \bmod n$ and -8 otherwise; write biases are zero; residual biases are 0 on the diagonal and -8 off-diagonal. This gives approximately one-hot reads, unit writes, and near-identity residual mixing, overriding Liger's default initialisation. Attention output and FFN down-projection weights are scaled by $1/\sqrt{n} = 1/2$ after the base initialisation with std 0.02. Biases and scalar gains receive no weight decay, while Φ uses 0.1. Embeddings are replicated across streams, which are averaged before the final RMSNorm and LM head.

Mixture-of-Depth Attention (MoDA) (Zhu et al., 2026). We use Pre-LN MoDA with the authors' `v17` Triton kernel, patched for our GPU types and head dimensions. RMSNorm precedes both attention and the FFN. Attention uses one softmax over causal sequence KV and earlier-layer, same-token depth KV, with scale $1/\sqrt{d_{\text{head}}}$. Attention KV projections are reused; FFN branches

[5] We follow the same recipe in AttnRes report (Team et al., 2026b) and Kimi K3 report (Team et al., 2026a) for choosing block size, i.e., fixing the number of blocks at approximately 8.

[6] `https://github.com/linkedin/Liger-Kernel/tree/v0.8.0`

add KV projections, except in the final block where no later layer reads them. The baseline's RoPE and additive residuals are retained.

Although the original paper favours Post-LN (Zhu et al., 2026), it optimises poorly under our shared recipe. As shown in Table 4, at $L = 24$ and learning rate 2×10^{-3}, Post-LN MoDA ends at validation CE 3.842 versus 2.764 for Pre-LN MoDA. Reducing the learning rate to 1×10^{-3} gives the best tested Post-LN MoDA result, 2.912, still above Pre-LN MoDA. We therefore retain Pre-LN MoDA for the shape comparison.

Table 4: **Validation loss for MoDA normalization variants ($L = 24$ model).**

Peak LR	5×10^{-3}	2×10^{-3}	1×10^{-3}	5×10^{-4}	2×10^{-4}
Pre-LN MoDA	2.775	**2.764**	2.773	2.864	–
Post-LN MoDA	4.893	3.842	**2.912**	2.933	3.443

Table 5: 400M model configurations across different width–depth aspect ratios.

Layers	Hidden	Intermediate	Heads	Head Dim.	Aspect Ratio	Backbone Size	Total Size	Total Diff.
16	1216	3248	16	76	76.00	284M	407M	$+0.28\%$
20	1120	2992	16	70	56.00	301M	414M	$+2.14\%$
24	1024	2736	16	64	42.67	302M	405M	0.00%
28	960	2560	16	60	34.29	310M	406M	$+0.21\%$
32	896	2400	16	56	28.00	309M	399M	-1.49%
42	800	2144	16	50	19.05	324M	404M	-0.31%
70	640	1712	16	40	9.14	345M	409M	$+0.94\%$

Table 6: Iso-backbone 300M model configurations

Layers	Hidden	Intermediate	Heads	Head Dim.	Aspect Ratio	Backbone Size	Backbone Diff.	Total Size
16	1248	3328	16	78	78.00	299M	-1.11%	425M
24	1024	2736	16	64	42.67	302M	0.00%	405M
32	896	2400	16	56	28.00	309M	$+2.26\%$	399M
42	768	2048	16	48	18.29	297M	-1.69%	375M
70	608	1632	16	38	8.69	312M	$+3.15\%$	373M
84	544	1456	16	34	6.48	299M	-0.99%	354M

C Complete Model Configurations

Tables 5, 7, 6, and 8 summarize the complete model configurations used in our experiments.

For the 400M benchmark, we use $(L, d) = (24, 1024)$ as the reference configuration. This is a convenient and representative Transformer shape: $d = 1024$ is a standard hidden dimension, gives a head dimension of 64 with 16 attention heads, and results in a model of approximately 405M parameters under our backbone. We construct the remaining shapes around this reference by trading width for depth, extending from the shallow–wide $(16, 1216)$ configuration to the substantially deeper $(70, 640)$ configuration. The reference shape also serves as the anchor for our iso-backbone study, making the two experimental settings directly comparable.

For the 1.6B experiments, we choose $(L, d) = (28, 2048)$ as the reference shape, matching the same Qwen3-1.7B (Yang et al., 2025) shape to define our larger-scale backbone. In particular, it uses $d_{\text{ff}} = 6144$, 16 query heads, 8 key–value heads, and head dimension 128. We then construct two deeper and narrower variants, $(40, 1728)$ and $(54, 1504)$, while keeping the total model size close to 1.6B. This gives three representative aspect ratios, 73.1, 43.2, and 27.9, covering a comparable shallow-to-deep range to the main 400M study.

The multi-scale 200M–500M suite is designed to preserve approximately the same three width–depth regimes across model scales. We use aspect ratios near 28, 43, and 76, corresponding to the

Table 7: 1.6B model configurations across different width–depth aspect ratios.

Layers	Hidden	Intermediate	Q/KV Heads	Head Dim.	Aspect Ratio	Backbone Size	Total Size	Total Diff.
28	2048	6144	16 / 8	128	73.14	1.409B	1.615B	0.00%
40	1728	5184	16 / 8	108	43.20	1.433B	1.607B	−0.51%
54	1504	4512	16 / 8	94	27.85	1.466B	1.617B	+0.11%

Table 8: Model configurations for the depth scaling ladder across 200M–500M scales.

Layers	Hidden	Intermediate	Heads	Head Dim.	Aspect Ratio	Backbone Size	Total Size	Total Diff.
200M – 4B training tokens								
24	672	1792	16	42	28.00	130M	198M	−3.42%
18	768	2048	16	48	42.67	127M	205M	0.00%
12	896	2400	16	56	74.67	116M	206M	+0.69%
300M – 6B training tokens								
28	800	2144	16	50	28.57	216M	296M	+1.09%
21	896	2400	16	56	42.67	203M	293M	0.00%
14	1056	2816	16	66	75.43	187M	294M	+0.18%
400M – 8B training tokens								
32	896	2400	16	56	28.00	309M	399M	−1.49%
24	1024	2736	16	64	42.67	302M	405M	0.00%
16	1216	3248	16	76	76.00	284M	406M	+0.28%
500M – 10B training tokens								
34	992	2656	16	62	29.18	403M	502M	−0.42%
26	1120	2992	16	70	43.08	392M	504M	0.00%
17	1344	3584	16	84	79.06	368M	504M	−0.16%

deep, intermediate, and shallow reference points in the 400M sweep. For example, at 400M these are exactly $(32, 896)$, $(24, 1024)$, and $(16, 1216)$. At other parameter scales, we adjust L and d to reproduce these aspect ratios as closely as possible while keeping the total parameter count matched within each scale. This construction allows us to test whether the preferred width–depth allocation changes with model size rather than with a particular absolute choice of width or depth.

D Potential Hidden Cost for mHC and Block AttnRes

Figure 1 shows that the scaling behavior of Full AttnRes and HC is less evident in their derived variants, Block AttnRes and mHC. We investigate the underlying causes of this discrepancy.

Block AttnRes. Block AttnRes restricts the sources for depth-mixing to a slower growing schedule. Layer outputs are accumulated into a running sum $\mathbf{s}_\ell$, and only at a *block boundary*, every $b/2$ layers for block size b, is the running sum appended to the set $\mathcal{B}_\ell$ of saved block outputs that later layers may attend over and then reset. At every layer the depth-mixing softmax is taken over $\{\mathbf{s}_\ell\} \cup \mathcal{B}_\ell$, so the mixed input is

$$\tilde{\mathbf{h}}_\ell \;\propto\; \alpha_\ell\, \mathbf{s}_\ell \;+\; \sum_{\mathbf{v} \in \mathcal{B}_\ell} \beta_{\ell,\mathbf{v}}\, \mathbf{v}, \qquad \alpha_\ell + \textstyle\sum_{\mathbf{v}} \beta_{\ell,\mathbf{v}} = 1, \tag{4}$$

where α_ℓ is the weight on the *last layer output* (the running sum) and the $\beta_{\ell,\mathbf{v}}$ are the weights on *previous block outputs*. We follow original AttnRes (Team et al., 2026b) and Kimi K3 (Team et al., 2026a) settings to fix the number of boundaries at 8, and the block size grows with depth ($b = 4, 5, 6, 7, 8$ for $L = 16, 20, 24, 28, 32$). Figure 7 shows how this schedule shapes the computation, by plotting α_ℓ for the pre-trained models.

Before the first block output has been saved, the depth-mixing softmax assigns essentially zero weight to the running sum: each layer's own computation is dropped from the mixed input, and the network effectively keeps re-reading the embedding. Only once saved block outputs exist does α_ℓ rise. The length of this initial dead segment scales with depth (2, 5, and 8 layers at $L = 16$, 24,

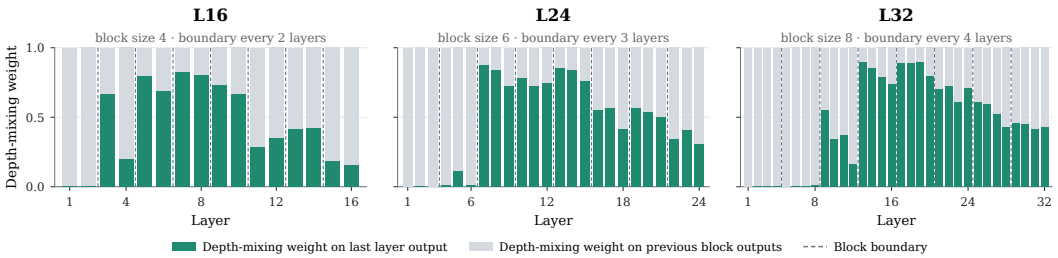

Figure 7: **Block AttnRes depth-mixing mechanism at $L = 16, 24, 32$.** At each layer, the share of the depth-mixing weight in Eq. 4 placed on the last layer output (α_ℓ, green) versus on previous block outputs ($\sum_{\mathbf{v}} \beta_{\ell,\mathbf{v}}$, grey); dashed lines mark block boundaries, where the set of saved block outputs is updated.

and 32), so a quarter of the $L = 32$ network can not benefit from stacking layers. This potentially explains why Block AttnRes stops benefiting from added depth.

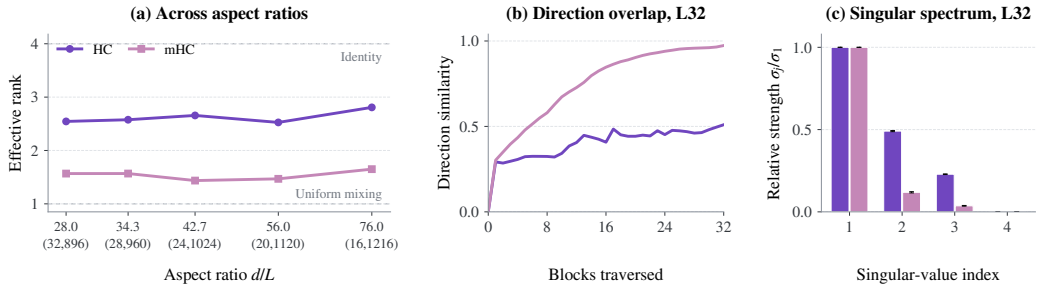

Figure 8: **Residual transport in HC and mHC. (a)** Effective rank across aspect ratios, with shapes (L, d) below each tick. Effective rank is the exponential of the entropy of the normalized singular values. **(b)** Mean absolute cosine similarity between propagated stream directions in $L = 32$ models. **(c)** Singular values of their full residual products, normalized by the largest. Statistics are averaged over sampled held-out FineWeb-Edu data; bands and error bars show 95% document-bootstrap intervals.

mHC. mHC retains HC's multi-stream structure but constrains its read/write weights to be non-negative and uses Sinkhorn–Knopp normalization to make residual mappings approximately doubly stochastic (Xie et al., 2025). These constraints aim to stabilize propagation across depth. However, as shown in Figure 1, mHC performs best at $L = 24$ and does not retain HC's gains in deeper, narrower models.

To examine this difference, we compose all attention and FFN residual maps along each forward pass. As shown in Figure 8(a), the resulting products have consistently lower effective rank in mHC: 1.44–1.65, compared with 2.53–2.81 for HC. At $L = 32$, mHC's propagated stream directions become nearly collinear, with much weaker non-leading singular values, as shown in Figure 8(b–c).

The doubly stochastic constraint suggests a mechanism for this concentration. Exact doubly stochastic maps share the uniform mixing matrix as a fixed point:

$$U = \frac{1}{m}\mathbf{1}\mathbf{1}^\top, \qquad A_\ell U = U A_\ell = U. \tag{5}$$

When repeated mixing contracts the remaining stream directions, residual products approach U, making earlier contributions less distinguishable across streams even as new branch outputs enter. Different sufficiently mixing orders of fixed residual maps then approach the same average, consistent with mHC's weaker permutation effects as shown in Figure 4. Together, these observations suggest a trade-off: stabilizing residual transport can reduce the diversity of long-range routes through which additional layers reuse earlier computations.

E FLOP ACCOUNTING

We report forward-pass FLOPs using the convention that one multiply–add counts as two FLOPs. The following accounting targets our sub-1B experimental setting, which uses standard multi-head attention (MHA), while the 1.6B models use GQA and therefore have a different attention-projection cost. We count the dominant matrix multiplications and vector dot products, while omitting secondary operations such as element-wise functions and RMSNorm. We follow the notation used throughout the paper: T is the sequence length, $L = n_{\text{layer}}$ is the number of Transformer layers, $d = d_{\text{model}}$ is the hidden dimension, d_{ff} is the FFN intermediate dimension, V is the vocabulary size, and m is the number of residual streams for HC and mHC. We use $m = 4$ in all experiments.

Pre-LN Transformer. For the baseline Transformer, we count the four attention projections Q, K, V, O, the three SwiGLU projections, the causal attention matrix products, and the LM head. The resulting cost is

$$F_{\text{Pre-LN}} = L\left(8Td^2 + 6Td\,d_{\text{ff}} + 2T^2d\right) + F_{\text{head}},$$

where

$$F_{\text{head}} = \begin{cases} 2TdV, & \text{for logits at all positions,} \\ 2dV, & \text{for final position logits.} \end{cases}$$

The $2T^2d$ term accounts for both $QK^\top$ and PV under causal attention. Each product would cost $2T^2d$ if evaluated densely, but we follow the convention that a causal kernel evaluates approximately half of the T^2 query–key pairs and that the two products together cost approximately $2T^2d$. A dense implementation that evaluates the full attention matrix would instead require approximately $4T^2d$.

HC. HC replaces the residual addition around each attention and FFN sublayer with a hyper-connection, giving $2L$ connectors. We count the dynamic routing projections and the read, residual mixing, and write operations, while omitting normalization and elementwise operations. The additional cost is

$$\Delta F_{\text{HC}} = 2L\left[2Tmd(m+1) + 2Tmd + 2Tm(m+1)d + 2Tmd\right].$$

For $m = 4$,

$$\Delta F_{\text{HC}} = 192\,TdL.$$

mHC. mHC likewise introduces one connector around each attention and FFN sub-layer. Its dominant additional cost is the projection of the flattened md-dimensional residual state to the $m^2 + 2m$ routing logits, followed by the read, $m \times m$ residual-stream mixing, and write operations. RMSNorm, sigmoid evaluations, and Sinkhorn–Knopp normalization (which is done entirely on-chip over very small matrices) are omitted under our FLOP convention. Thus,

$$\Delta F_{\text{mHC}} = 2L\left[2Tmd(m^2 + 2m) + 2Tmd + 2Tm^2d + 2Tmd\right].$$

For $m = 4$,

$$\Delta F_{\text{mHC}} = 480\,TdL.$$

The higher FLOP cost of mHC relative to HC is therefore dominated by its $md \to m^2 + 2m$ routing projection rather than by the Sinkhorn iterations.

Full AttnRes. Full AttnRes scores each preceding residual source using the query–source dot product $w^\top v_i$ and forms a weighted sum of the source vectors. For each source, these two operations cost $2Td$ FLOPs each, giving $4Td$ FLOPs per source. Source normalization and the softmax over depth are omitted. The first attention sub-layer reads the embedding directly and requires no depth-mixing operation. The remaining sub-layers and the LM-head readout mix $2, 3, \ldots, 2L+1$ sources, respectively. Hence,

$$\Delta F_{\text{Full}} = 4Td \sum_{\ell=2}^{2L+1} \ell = 4TdL(2L+3),$$

and we further approximate for simplicity

$$\Delta F_{\text{Full}} \approx 8TdL^2.$$

Block AttnRes. Block AttnRes uses the same depth-mixing operation but accumulates sub-layer outputs within blocks. Following our experimental setup, the $2L$ attention and FFN sub-layers are divided into eight blocks (more blocks are possible, but here we use 8 following their recommendations) with block size

$$b = \frac{2L}{8}.$$

Indexing the sub-layers by $k = 0, \ldots, 2L-1$, the first attention sub-layer ($k = 0$) reads the embedding directly and requires no depth-mixing operation. For $k = 1, \ldots, 2L-1$, the mixture contains the embedding, the current running block sum, and one additional frozen source for every completed block, giving

$$2 + \left\lfloor \frac{k}{b} \right\rfloor$$

sources. The LM head mixes the embedding with the eight completed block outputs, giving nine sources. The additional cost is therefore

$$\Delta F_{\text{Block}} = 4Td \left[\sum_{k=1}^{2L-1} \left(2 + \left\lfloor \frac{k}{b} \right\rfloor \right) + 9 \right], \qquad b = \frac{2L}{8}.$$

F Prefill Cost Measurement

We complement the analytical FLOP accounting in Appendix E with hardware measurements of prefill cost. Figure 9 reports results at sequence length $T = 4096$ for our 400M models in BF16 on A100-SXM4-40GB GPUs. We measure forward FLOPs and KV-cache size per sequence, as well as peak memory and GPU-hours for a batch size of 8.

A first observation is that **depth itself introduces a non-negligible systems cost even at a fixed parameter scale**. As the aspect ratio decreases from 76.0 to 9.1, prefill FLOPs increase from roughly 3.0 to 4.4 TFLOPs per sequence, while the KV cache increases by more than $2\times$. The latter is identical across architectures, confirming that this component is intrinsic to the width–depth reallocation rather than to a particular residual design. More importantly, **the practical cost of depth grows substantially fast**. Peak memory and GPU-hours increase sharply toward the deepest configurations. Full AttnRes has the largest peak-memory footprint, consistent with its explicit cross-layer source access, whereas HC shows the highest measured GPU-hour cost despite having a much smaller difference in nominal FLOPs.

These measurements reveal a clear accuracy–efficiency trade-off of depth scaling. For Pre-LN, deeper–narrower shapes incur higher system cost without improving validation loss. In contrast, HC and Full AttnRes obtain better modeling performance in this regime, but at increased memory and GPU-hour cost. We do not perform dedicated system optimization for deep–narrow architectures in these measurements. Realizing the gains of depth scaling requires architecture–system co-design.

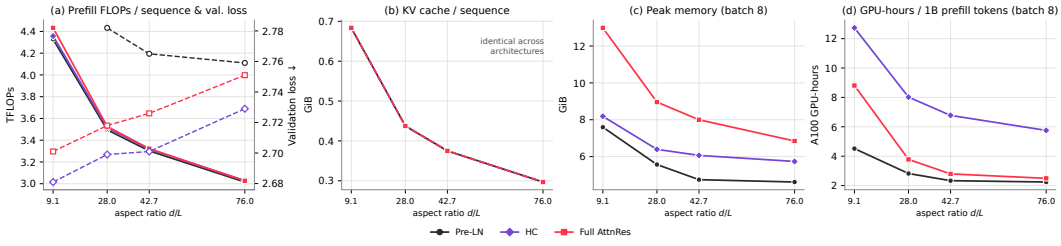

Figure 9: **Measured prefill cost** including FLOPs, KV cache memory, Peak GPU memory and GPU hours at sequence length $= 4096$, model size at about 400M, BF16, A100-SXM4-40GB. The dashed line in (a) corresponds to the validation loss.

G Complete Results of Causal Score and Permutation Score

This section reports the complete layer-pair heatmaps of causal score and permutation score in Section 4.1, as shown in Figure 10, 11.

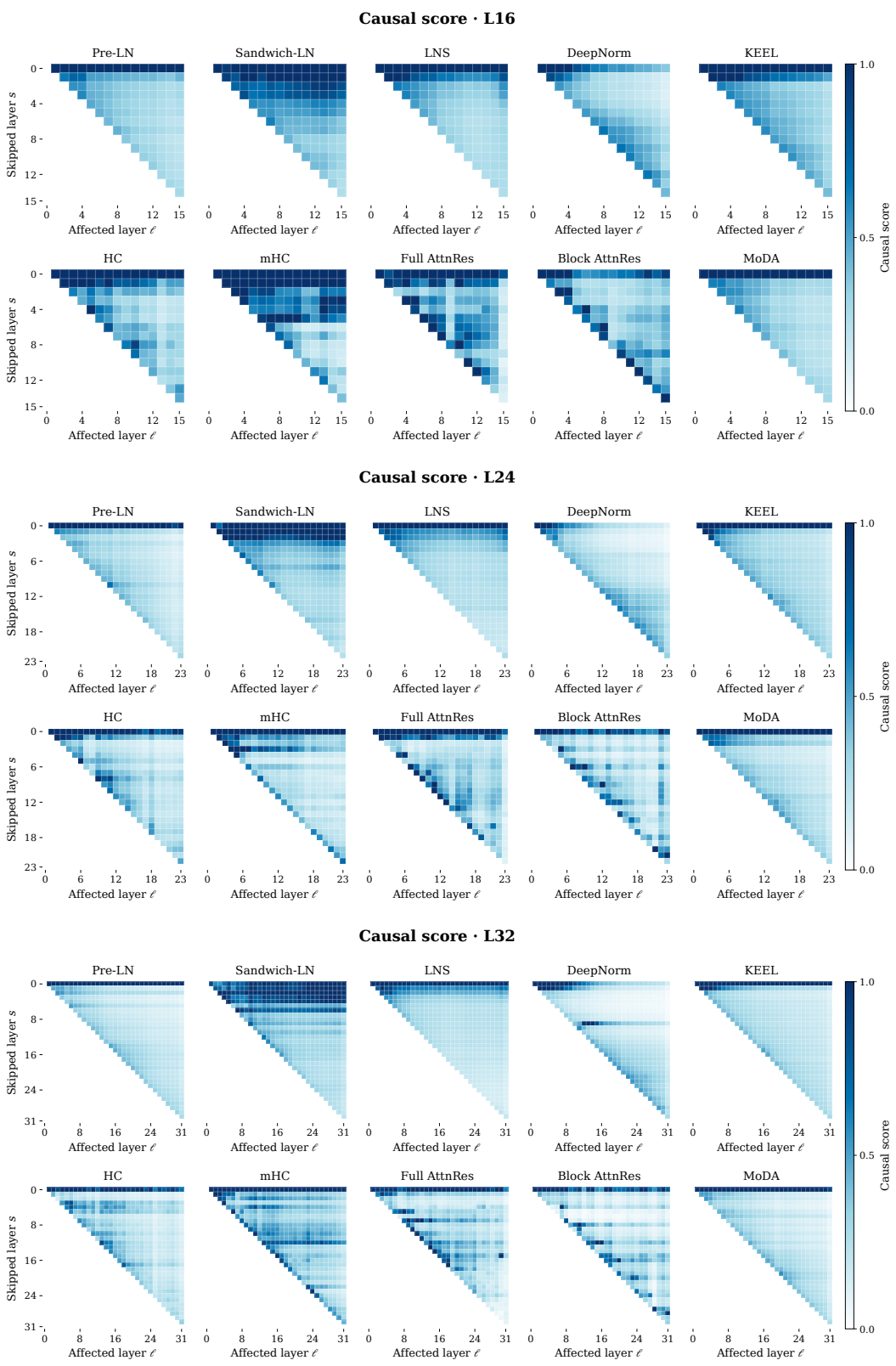

Figure 10: Causal score for all architectures at $L = 16$, 24, and 32 (top to bottom).

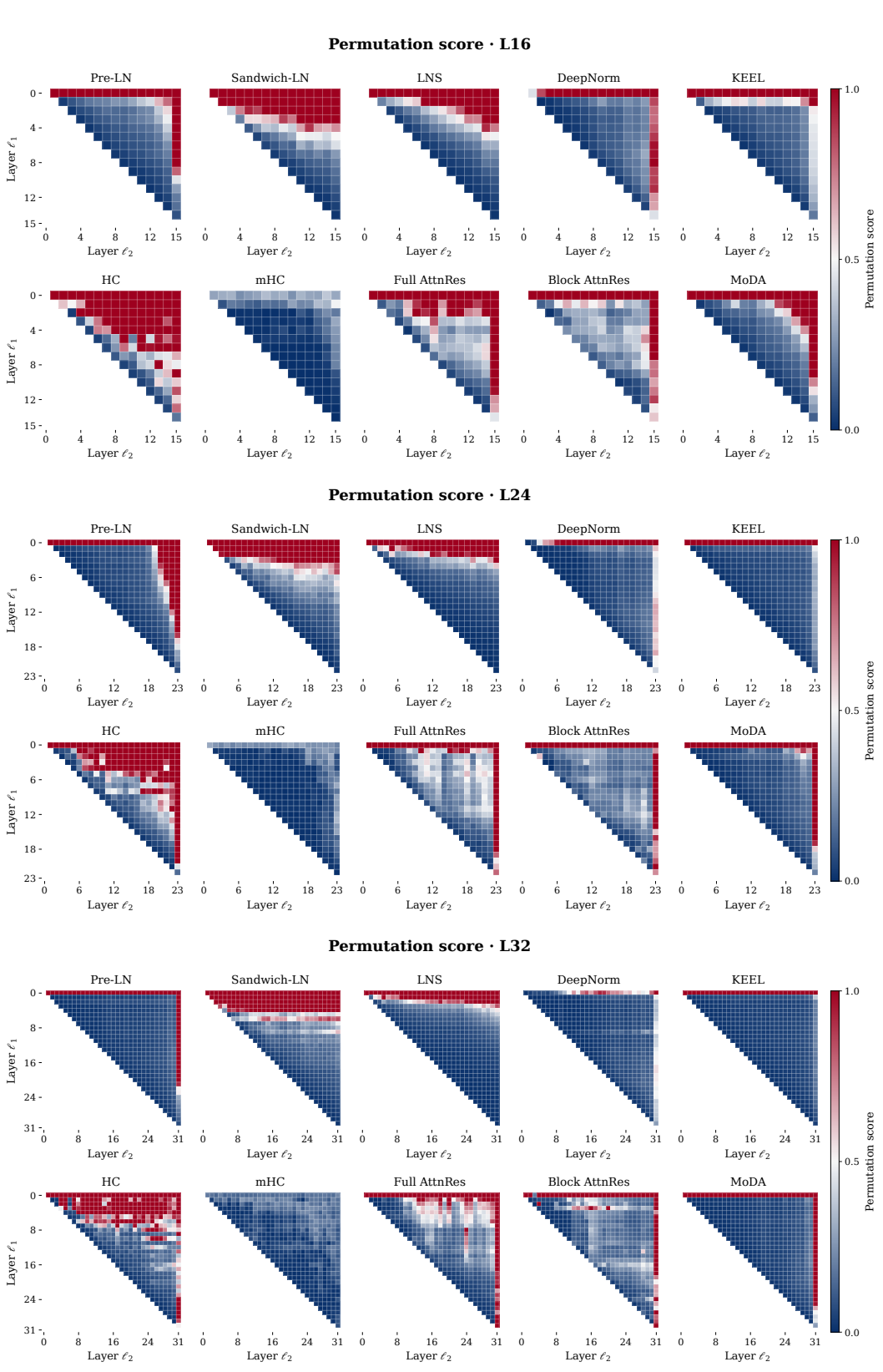

Figure 11: Permutation score for all architectures at $L = 16$, 24, and 32 (top to bottom).