Variable-Width Transformers

Zhaofeng Wu[MIT] **Oliver Sieberling**[MIT] **Shawn Tan**[MIT-IBM]
Rameswar Panda[MIT-IBM] **Yury Polyanskiy**[MIT] **Yoon Kim**[MIT]
[MIT] MIT [MIT-IBM] MIT-IBM Watson AI Lab
zfw@csail.mit.edu

Abstract

Scaling model size, specifically depth and width, has driven significant progress in transformer-based language models. However, most architectures maintain a constant width across all layers, allocating a fixed parameter and computation budget evenly despite different layers potentially playing distinct computational roles. In this work, we empirically investigate nonuniform capacity allocation across network depth by proposing a $\times$-shaped ><former architecture. This design maintains wider early and late layers while narrowing the middle layers, utilizing a parameter-free residual resizing mechanism. Across decoder-only language models ranging from 200M to 2B parameters (dense) and 3B parameters (MoE), our ><former consistently outperforms parameter-matched uniform baselines on language modeling loss. By reducing the average layer width, this architecture also requires fewer overall FLOPs (22% reduction under fitted loss-matched scaling curves) and smaller KV cache memory and I/O cost (15% reduction). In analysis, we show that this bottleneck structure results in qualitatively different representations in residual streams. Overall, our results demonstrate that nonuniform width allocation can result in more resource-optimal scaling of language models.

Figure 1: We propose ><former, where different layers have different widths. We specifically employ a $\times$-shaped architecture, where inactive dimensions are copied upward in the residual stream. We find that this improves performance and saves on training FLOPs, KV cache memory, and I/O cost.

1 Introduction

Scaling has been a critical driver of progress in modern AI. One major axis of scaling is model size. In the context of transformers, model size is a function of the dimension of the transformer block (a

We release our code at https://github.com/ZhaofengWu/variable-width-transformers.

model's "width")[1] and the number of transformer blocks (a model's "depth"). Consequently, much prior work has investigated how to optimally scale model size through scaling transformer width and depth. While early research suggested that a model's shape (i.e., the ratio of width to depth) mattered less than the total parameter count for performance (Kaplan et al., 2020), subsequent work found that model shape can lead to nontrivial differences (Levine et al., 2020; Tay et al., 2021; Petty et al., 2024) and should be taken into account when fitting scaling laws (McLeish et al., 2025).

Yet, even as the optimal global shape of transformers is debated, these studies generally preserve a less-examined assumption: a model's width is constant across depth. That is, once a hidden dimension size is chosen, every transformer block receives approximately the same computation/parameter budget. This uniform-width design is convenient, but not obviously optimal. Different layers may play different roles during computation, and a fixed total parameter or FLOP budget need not be allocated evenly across depth. This motivates a general question: under a fixed depth and parameter budget, should all layers have the same width, or should capacity be distributed nonuniformly?

We study this question empirically by training decoder-only transformer language models (LMs) with nonuniform allocation of parameters and compute across depth. Concretely, for a given parameter and depth constraint, we vary the model shape across several settings: growing ($\vee$-shaped), narrowing ($\wedge$-shaped), growing then narrowing ($\diamond$-shaped), and narrowing then growing ($\times$-shaped). Across these settings, we find that $\times$-shaped models (wide in early and late layers but narrower in the middle) outperform parameter-matched constant-width transformers. We call them > <formers. This differs from prior work on layerwise allocation of only FFN intermediate dimensions, which found benefits from allocating more computation to middle layers (Ikeda et al., 2025); our results instead suggest that reallocating the full block width leads to a different optimal profile.

A key implementation detail is how variable-width layers interact with the residual stream. Naïvely changing the residual dimension between layers introduces projection bottlenecks and changes the skip path. We instead keep a fixed global residual dimension and allow each block to read from and write to a layer-specific slice of the residual stream. Coordinates not used by a given block bypass that block and are projected upstream via copying. We find that this fixed-residual construction is important for realizing the gains from nonuniform width profiles.

Nonuniform width allocation also has efficiency benefits: it requires fewer training and inference FLOPs than constant-width transformers, while also reducing the KV cache memory and I/O cost for moving activations. Because a layer's parameter count scales quadratically with its width, while attention FLOPs and KV cache size scale linearly, matching the parameter count of a uniform baseline results in a reduction in average layer width (and hence the KV cache). Across models with 200M–2B parameters, > <formers achieve approximately a 3% relative improvement in perplexity over parameter-matched constant-width baselines while reducing KV-cache size by about 10% and FLOPs by about 3%. These benefits also extend to mixture-of-experts transformers. We further analyze how the optimal bottleneck width and bottleneck location depend on the model budget, providing empirical guidance for scaling nonuniform-width transformers. Finally, we also perform analyses to understand the benefit of > <former, showing that it employs a different representation strategy than the constant-width baseline and mitigates mid-layer representation collapse.

2 Variable-Width Transformers

A standard transformer contains a series of L layers. In each layer $\ell \in [1, L]$, a transformer block $\mathcal{B}^\ell : \mathbb{R}^d \to \mathbb{R}^d$ transforms the input from the previous layer $\mathbf{x}^{\ell-1}$ by $\mathbf{x}^\ell = \mathcal{B}^\ell(\mathbf{x}^{\ell-1}) + \mathbf{x}^{\ell-1}$. d is the model dimension. We define $\mathbf{x}^0$ as the input embeddings.

In this work, we question why d must be held constant. Much past work has shown that different layers of a transformer LM perform distinct functions, which naturally may require different amounts of capacity (Tenney et al., 2019; Meng et al., 2022; Sajjad et al., 2023; *i.a.*). This motivates each layer ℓ having a different dimension d_ℓ.

One practical challenge, however, is that this requires resizing between layers: $\mathbf{x}^\ell = \mathcal{B}^\ell(f^\ell(\mathbf{x}^{\ell-1})) + f^\ell(\mathbf{x}^{\ell-1})$ where $f^\ell : \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell}$ resizes the hidden state. We consider a parameter-free approach. When shrinking dimensions, i.e., $d_\ell < d_{\ell-1}$, we simply truncate the extra dimensions, i.e., $f^\ell(\mathbf{x}) = \mathbf{x}[:d_\ell]$. When expanding dimensions, i.e., $d_\ell > d_{\ell-1}$, we restore each previously truncated dimension

[1] The intermediate MLP dimension is typically a fixed multiple of the model dimension (though see Ikeda et al., 2025).

from the most recent layer that actively processed it. Formally, for each coordinate index $i \in \{1, \ldots, d_\ell\}$, the i-th element of the resized hidden state $f^\ell(\mathbf{x}^{\ell-1}) \in \mathbb{R}^{d_\ell}$ is constructed as:

$$[f^\ell(\mathbf{x}^{\ell-1})]_i = [\mathbf{x}^{\ell'}]_i \quad \text{where} \quad \ell' = \max\{\tilde{\ell} < \ell \mid d_{\tilde{\ell}} \geq i\} \tag{1}$$

If no such prior layer exists (i.e., if the required dimension exceeds the maximum width of all preceding layers), the coordinate is padded with 0. See §4.4 for ablations that show that these methods outperform alternatives such as training a projection layer or always padding with 0s.

Using this expansion method, we can mathematically conceptualize our variable-width model as a uniform-width model except (1) each layer only reads from/writes to a subset of residual stream dimensions, and (2) it has a larger residual stream width (equal to the width of the widest layer).

We investigate different shapes with two additional parameters, ℓ^*, the layer index of a bottleneck layer, and d_{ℓ^*}, its dimension. We parameterize the rest of the layer widths geometrically:[2] $d_\ell = \alpha^- d_{\ell-1}$ with change rate α^- for $\ell \leq \ell^*$ (the early layers) and $d_\ell = \alpha^+ d_{\ell-1}$ with change rate α^+ for $\ell > \ell^*$ (the late layers). With different settings of α^+, α^-, and ℓ^*, we recover different kinds of shapes: when $\alpha^- < 1$ and $\alpha^+ > 1$, we obtain a $\times$-shaped model; when $\alpha^- > 1$ and $\alpha^+ < 1$, we obtain a $\diamond$-shaped model; when ℓ^* is 1 or L, we obtain a $\vee$ or $\wedge$-shaped model. We keep the size of the input and output embeddings unchanged: the QKV projections of the first layer and the MLP down-projection of the last layer adjust for these size mismatches.[3]

A compelling property of our variable-width architecture is that, when matched in parameter count to a constant-width baseline, it requires strictly fewer overall FLOPs and has a strictly lower average layer width (and thus, lower KV cache size and lower I/O cost for moving activations). Concretely, the parameter count of a transformer layer is dominated by the linear projection matrices (QKV, output, and MLP), which scale quadratically with the hidden dimension: $P_\ell \approx K d_\ell^2$, where K is a constant depending on the number of projection matrices and the MLP expansion factor. Therefore, if we match the parameter count of a variable-width model to a baseline of constant width d, we effectively equate the sum of the squared dimensions:

$$K \sum_{\ell=1}^{L} d_\ell^2 = KLd^2 \implies \frac{1}{L} \sum_{\ell=1}^{L} d_\ell^2 = d^2.$$

Because the square of the mean is upper-bounded by the mean of the squares, and because variable-width ensures the widths d_ℓ are not constant, the average layer size is strictly smaller:

$$\left(\frac{1}{L} \sum_{\ell=1}^{L} d_\ell \right)^2 < \frac{1}{L} \sum_{\ell=1}^{L} d_\ell^2 = d^2 \implies \frac{1}{L} \sum_{\ell=1}^{L} d_\ell < d.$$

For FLOPs, first of all, the number of FLOPs per token in a linear projection is strictly proportional to the number of weights, and so when parameter-matched, the total dense FLOPs remain identical to the baseline. For attention dot-products, their FLOPs scale linearly with the hidden dimension: $\text{FLOPs}_\ell \propto N^2 d_\ell$, where N is the sequence length. Therefore, the total attention compute $\sum_{\ell=1}^{L} N^2 d_\ell = N^2 \sum_{\ell=1}^{L} d_\ell$ is consequently strictly lower than the baseline $N^2 Ld$.[4]

In summary, there are 4 parameters for a variable-width transformer, ℓ^*, d_{ℓ^*}, α^+, and α^-. We set two constraints: $d_1 = d_L$ (for $\diamond$ and $\times$ shapes) and that the parameter count matches a constant-width baseline.[5] So for our experiments, we consider ℓ^* and d_{ℓ^*} as two hyperparameters, and automatically solve for all layer widths.[6] See §A for our derivation.

3 > <former

In this section, we discuss training variable-width transformers. After introducing our training setup, we first establish that a $\times$-shaped model works best, and then identify a parameterization of the

[2] A geometric layer width schedule outperforms an arithmetic one in preliminary experiments.

[3] The residual connection in the final layer MLP is truncated accordingly.

[4] In practice, adjusting for the sizes of the input and output embeddings (see above; formalized in §A) introduces a minor parameter correction. Nevertheless, this $O(1)$ boundary effect is heavily dwarfed by the $O(L)$ bulk layer parameters, preserving these theoretical results under meaningful width schedules.

[5] As shown in the proof, we cannot match the parameter count and the FLOP count at the same time.

[6] After solving the layer widths, we post-hoc round each width to the nearest 32, which is the number of attention heads we use (16) times 2 (for RoPE (Su et al., 2024)).

Table 1: Pre-training hyperparameters.

Parameters	Layers (L)	Hidden (d)	Batch Size	Tokens	Experts (Tot/Act)	MLP Interm. Size
Dense Models						
200M	16	640	512	10B	–	–
500M	24	960	1024	25B	–	–
1B	32	1280	2048	50B	–	–
2B	40	1600	4096	100B	–	–
Mixture of Experts						
3B (1B active)	40	1600	4096	100B	22 / 3	512

bottleneck layer index ℓ^* and dimension d_{ℓ^*} that works well across model sizes. Using this recipe, we pre-train > <formers and constant-width baselines across sizes and find that > <formers consistently achieve better loss and downstream task performance with a smaller pre-training FLOPs footprint and KV cache size.

3.1 Training Setup

We pre-train four model sizes—200M, 500M, 1B, and 2B—with different numbers of layers and hidden sizes (Table 1). For each, we pre-train constant-width transformers and variable-width models. We also consider a Mixture-of-Experts (MoE) model with 3B total/1B active parameters. For parameter-matching the variable-width MoE model with the baseline, we match the number of total parameters—this results in the variable-width model having 3% fewer active parameters, but we show in §3.4 that it still outperforms the constant-width baseline despite this.

For pre-training data, we train on DCLM (Li et al., 2024). For each model size, we train models to 2.5× Chinchilla-optimal (Hoffmann et al., 2022), i.e., the number of trained tokens is equal to 50 times the parameter count, e.g., 100B tokens for the 2B model. We train on length-4096 sequences, with model-size-dependent batch sizes (Table 1). Inputs are tokenized with OpenAI's `cl100k_base`.[7]

All models are trained with maximal update parametrization (μP; Yang et al., 2024). We use μP-aware initialization and optimizer parameter groups, with the same AdamW hyperparameters across scales: learning rate 10^{-2}, $\beta = (0.9, 0.95)$, weight decay 0.1, and $\epsilon = 10^{-10}$. We use a power learning-rate decay schedule. The learning rate is linearly warmed up for approximately the first 8% of training steps and then decayed for the remaining steps. All models are trained in bfloat16 precision. Following common practice, we omit bias terms from all linear projections, including attention, MLP, and output projections (Chowdhery et al., 2022; Groeneveld et al., 2024; *i.a.*). We use the SwiGLU activation (Shazeer, 2020).

We measure the training loss of each model,[8] averaged over the final 1,000 steps (with a 10-step increment) for smoothing. We also report pre-training FLOPs in PFLOP/s-days, the number of days required assuming 1 PFLOP per second (Team et al., 2026). Finally, we compare the average layer size, which is proportional to the KV cache size during inference.

3.2 The × Shape Works Best

We first explore different shapes on the 500M-parameter scale. Beyond a regular constant-width transformer, we experiment with a ◇ shape, a × shape, a ∨ shape, and a ∧ shape. Due to the optimal hyperparameter (ℓ^* and d_{ℓ^*}, §2) potentially differing per shape, we experiment with 3 choices for each hyperparameter per shape. This amounts to 9 runs for the × shape and the ◇ shape, and 3 runs for the ∨ shape and the ∧ shape, for which ℓ^* is irrelevant. In Figure 2, we plot their loss and the pre-training FLOPs. We see that the × shape consistently performs the best.[9]

[7] https://github.com/openai/tiktoken

[8] Because we do not repeat any data, this is in expectation the same as held-out loss.

[9] Anecdotally, our initial intuition was to pursue a ◇-shaped model, increasing the computation in middle layers which are often associated with semantic computations (Tenney et al., 2019). We nevertheless proceed with ×-shaped models due to these empirical results.

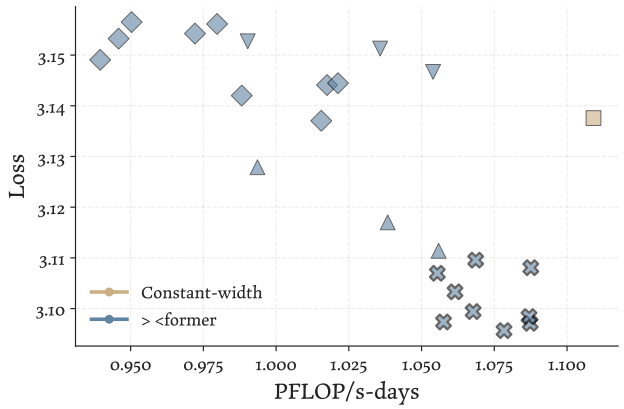

Figure 2: Comparing variable-width transformers with different shapes, each sweeping over multiple hyperparameter choices. **The $\times$-shaped model performs the best.**

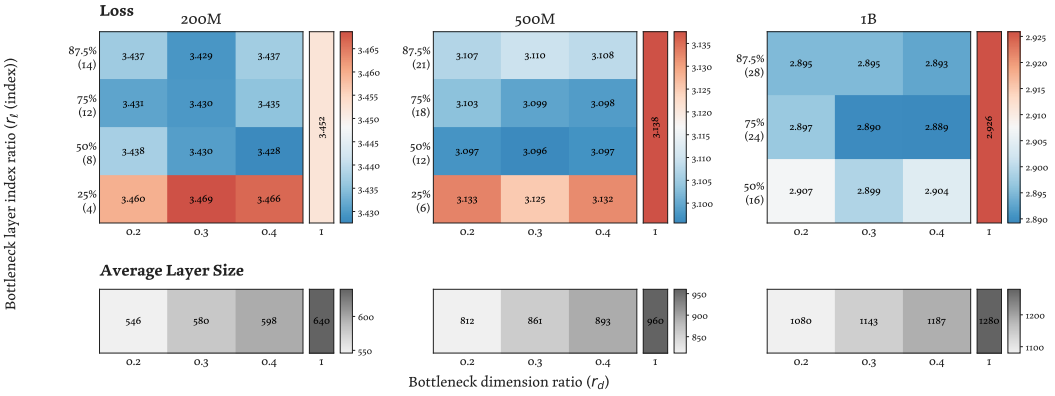

Figure 3: The effect of the bottleneck layer index and dimension on language modeling loss, parameterized as a ratio to the total number of layers and the base dimension. $\ell^* = r_\ell L$ and $d_{\ell *} = r_d d$. We also show the baseline performance, indicated using $r_d = 1$, and the resulting average layer size. **This parameterization yields a relatively consistent model performance pattern across model sizes.**

3.3 Finding a Specific Width Schedule

As mentioned in §2, we need to choose a bottleneck layer index ℓ^* and the bottleneck dimension d_{l^*}. Ideally, we want to find a recipe that works well across model sizes so that we do not have to search for them individually at each model size. Therefore, we parameterize these two hyperparameters as ratios to the total number of layers L and the hidden size d: $\ell^* = r_\ell L$ and $d_{l^*} = r_d d$. We sweep over different values of r_ℓ and r_d at small model sizes: 200M, 500M, and 1B. Figure 3 shows the results. While it is not the case that a single (r_ℓ, r_d) pair is consistently the best, the fact that such a ratio-based parameterization leads to *roughly* similar trends across model sizes is interesting. By default, based on this sweep, we use $\ell^* = 0.75L$ and $d_{l^*} = 0.3d$ going forward.

3.4 > <former Outperforms Constant-Width Transformer

Table 2 shows, at all model sizes that we tested, > <former outperforms the constant-width transformer, while requiring fewer FLOPs and average layer size (i.e., with a reduction in KV cache size).

In Figure 4 (left), we fit a scaling law curve on loss vs. pre-training FLOPs to > <formers and constant-width transformers (Kaplan et al., 2020), finding a tight fit. Similarly, in Figure 4 (right), we also find a tight power-law fit on loss vs. the average layer size. From these scaling law curves, we compute that > <former can achieve the 2B constant-width transformer's loss (2.751) with 77.8%

Table 2: The performance, pre-training FLOPs, and average layer size of > <formers vs. constant-width transformers. **> <former consistently achieves lower loss with lower pre-training FLOPs and average layer size.**

Size	Model	Loss	PFLOP/s-days	Avg layer size
200M	Transformer	3.452	0.18	640
	> <former	**3.430**	**0.17 (−3.2%)**	**576 (−10.0%)**
500M	Transformer	3.138	1.11	960
	> <former	**3.099**	**1.07 (−3.7%)**	**855 (−11.0%)**
1B	Transformer	2.926	4.52	1280
	> <former	**2.890**	**4.41 (−2.6%)**	**1145 (−10.5%)**
2B	Transformer	2.751	16.92	1600
	> <former	**2.726**	**16.49 (−2.5%)**	**1426 (−10.9%)**
3B/1B MoE	Transformer	2.726	10.13	1600
	> <former	**2.710**	**9.66 (−4.6%)**	**1426 (−10.9%)**

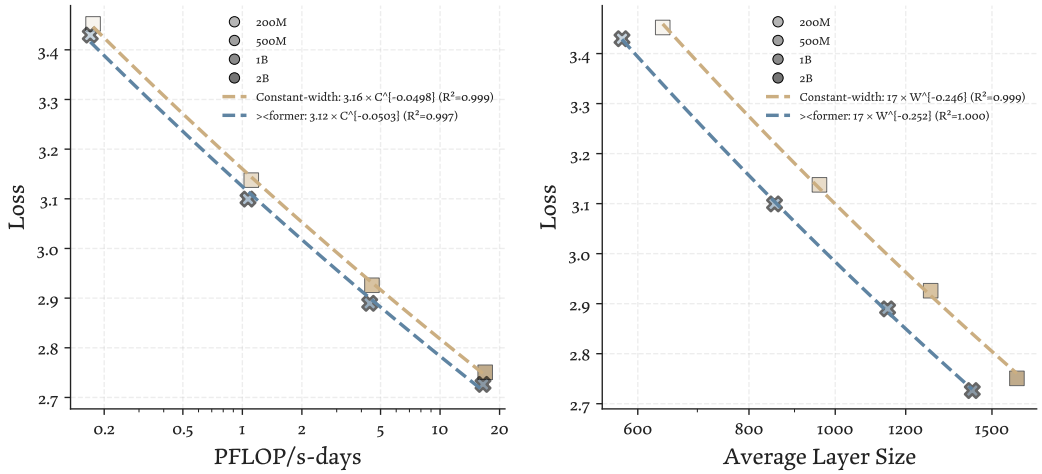

Figure 4: Language modeling loss vs. pre-training FLOPs (left) and average layer size (right). **> <former produces lower loss at smaller FLOP and average layer size costs.**

FLOPs and 85.1% average layer width. Furthermore, both scaling law curves show that not only does > <former have a smaller intercept, but it has a slightly steeper scaling exponent too, suggesting that the gaps might widen at larger sizes.

We also test these models on standard LM downstream evaluation benchmarks using the `lm-evaluation-harness` (Gao et al., 2024) in the zero-shot setting. This suite covers natural language understanding (NLU) tasks such as common-sense reasoning, reading comprehension, etc., as well as perplexity-based tasks. For multiple-choice tasks, we report normalized accuracy when available, since it corrects for answer-length effects by normalizing choice likelihoods. When it is not provided, we report standard accuracy. We report the dataset statistics and metrics in Table 5 in §B. We evaluate the 2B models and the MoE models and show the results in Table 3. > <formers consistently outperform constant-width transformers on perplexity-based tasks. The 2B > <former also leads on most NLU tasks. The MoE > <former is mixed on NLU accuracy but improves both perplexity metrics; at these model sizes, we treat perplexity as the more informative metric of LM quality. We again note that > <former achieves this with fewer FLOPs and memory, and also fewer active parameters for the MoE model (§3.1).

Model	Accuracy (↑)												Perplexity (↓)	
	ARC-C	ARC-E	BoolQ	COPA	HellaSwag	LAMBADA	OBQA	PIQA	RACE	SciQ	WinoGrande	Avg.	LAMBADA	WikiText
2B constant-width	33.0	59.5	59.4	76.0	55.9	55.4	33.8	73.3	34.3	79.5	57.0	56.1	8.18	16.96
2B > <former	34.4	**63.3**	60.9	73.0	**57.9**	56.1	33.6	74.4	33.4	82.0	**60.2**	**57.2**	**7.43**	**16.32**
MoE baseline	33.7	62.2	**63.0**	77.0	57.3	56.1	37.4	74.8	34.6	**83.9**	55.6	57.8	7.78	16.36
MoE > <former	33.2	61.0	59.5	80.0	**58.7**	56.2	37.8	75.2	34.3	80.2	**60.1**	57.8	**7.45**	**15.98**

Table 3: Model performance on standard LM evaluation datasets. For accuracy metrics, bold indicates the higher value when significant at $p < 0.05$ under a one-sided test; for perplexity, bold indicates the lower value. **> <formers consistently outperform constant-width transformers on perplexity-based tasks, and the 2B > <former wins on most natural language understanding tasks.**

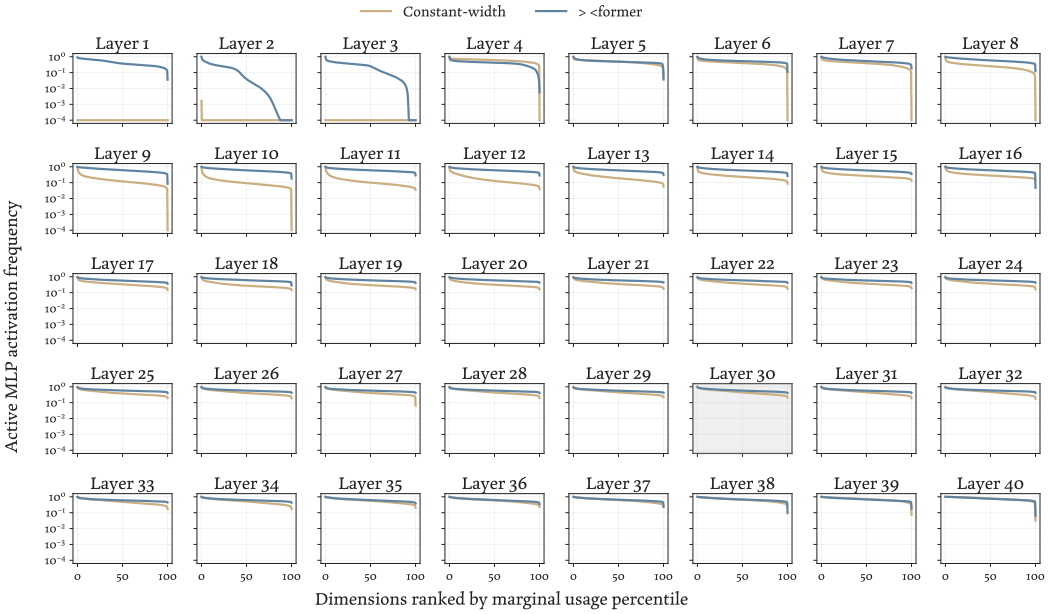

Figure 5: The utilization frequency of MLP activation dimensions in the 2B > <former vs. the 2B constant-width transformer, visualized separately for each layer. The shaded panel corresponds to the bottleneck layer. **> <former more evenly utilizes MLP activation dimensions.**

4 Analysis

Transformers are known to use depths inefficiently (Gromov et al., 2025), frequently developing "compression valleys" where their middle layers collapse in representational capacity and compress computations (Skean et al., 2025; de Llano et al., 2026). By inspecting both MLP intermediate activations and the residual stream after each layer, we find that > <former employs a different representation strategy, where it mitigates the collapse in middle layers and more effectively uses its capacity than a constant-width transformer.

4.1 > <former Improves MLP Activation Utilization

Intuitively, > <former enforces an information bottleneck that may encourage the model to more effectively use its representation capacity. We operationalize this by inspecting the utilization of intermediate MLP activations. Prior interpretability work has viewed a transformer MLP layer as containing key-value memories: the up-projection layer encodes keys for distinct concepts, the intermediate activation represents the MLP input's affinities with the concepts, and the down-projection encodes values for those concepts, taking a linear combination of them weighted by the affinity scores (the activation) (Geva et al., 2021, 2022). We measure the MLP activation density of the 2B > <former vs. the constant-width transformer on the WikiText-2 validation split with 252,986 tokens (Merity et al., 2017). Because SwiGLU is continuous, we consider a dimension as active iff its activation magnitude is larger than a certain threshold. In Figure 6, we show that > <former enforces denser activations within MLPs across thresholds.

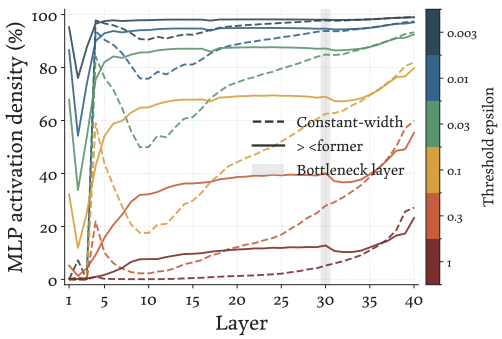

Figure 6: The density of MLP activations in the 2B > <former vs. the 2B constant-width transformer, across thresholds. **> <former consistently more densely activates MLP activation dimensions.**

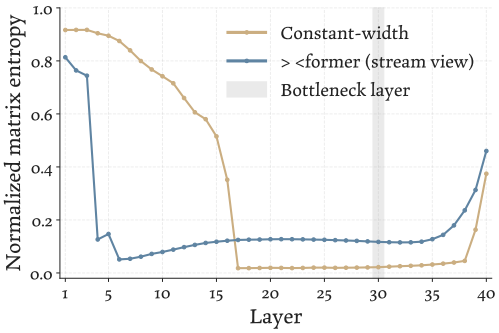

Figure 7: The normalized matrix entropy (§4.2) of layer outputs in the 2B > <former vs. the 2B constant-width transformer. **> <former has a higher matrix entropy in middle-to-final layers, which corresponds to more even usage of the residual dimensions in those layers.**

Dense activation are not necessarily desirable, so we also inspect the marginal utilization of each MLP activation dimension: how often a dimension is activated across tokens (thresholded at 0.1). In the mechanistic interpretability literature, low marginal utilization and "dead" dimensions are strong indicators of under-utilized capacity (Bricken et al., 2023; Gao et al., 2025). Figure 5 shows this quantity across layers. While neither model has a perfectly even distribution, > <former consistently achieves substantially better load-balancing between activation dimensions. In §C, we also show a similar trend if we additionally take activation magnitude into account, demonstrating that > <former more evenly uses the activation dimensions in the middle layers of the network.

4.2 > <former Mitigates Middle-layer Representation Collapse

We now turn from studying MLP activations to the residual stream after each layer. Recent analyses of deep, constant-width LMs reveal the emergence of "compression valleys," where the LM's middle layers collapse in representational capacity, characterized by a severe drop in representational entropy (Skean et al., 2025; de Llano et al., 2026). Following de Llano et al. (2026), we track the normalized matrix entropy of the residual stream across all layers:

$$\frac{1}{\log r}\left(-\sum_{j=1}^{r} p_j \log p_j\right), \quad p_j = \sigma_j^2 / \|\mathbf{X}\|_F^2 \tag{2}$$

where σ_j are the sorted singular values of the input-feature representation matrix $\mathbf{X}$ with rank r, again computed using the WikiText-2 validation split. Closely related to the effective dimension metric (Hill, 1973; Roy & Vetterli, 2007), a higher matrix entropy indicates a more "even" use of the representation space.

We consider the per-layer hidden states in this analysis. For > <former, recall our interpretation from §2 that considers it having a wide residual stream where each layer reads from/writes to only a subset of dimensions. Accordingly, we consider this wide residual stream as its effective hidden states.

In Figure 7, we see that the baseline model exhibits a severe compression valley: in middle layers, its normalized entropy drops to near-zero, indicating that the token representations have collapsed into a highly degenerate, low-rank subspace despite the large width. This is consistent with prior findings (Skean et al., 2025; de Llano et al., 2026). In contrast, > <former restructures this dynamic. While it actively lowers its entropy in the early layers to compress the representation (anticipating the width reduction), it avoids the middle-layer collapse. Throughout the bottleneck and final layers, > <former maintains a higher normalized entropy, potentially suggesting that physically constraining the parameter space encourages the network to maintain a high-entropy manifold.

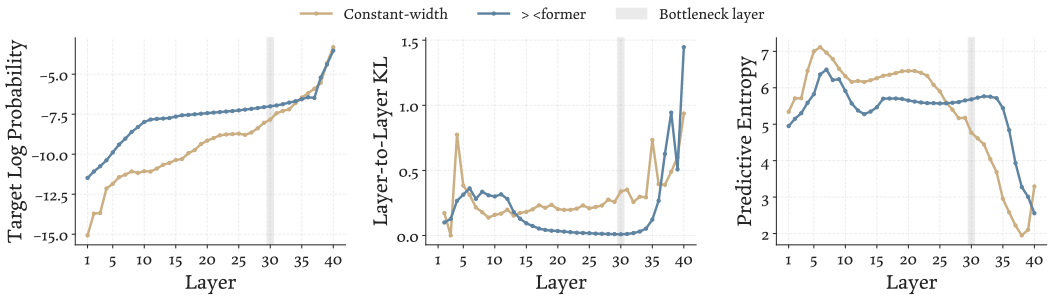

Figure 8: Logit lens analysis of the 2B > <former versus the constant-width baseline. **Left:** > <former assigns higher target-token probability through much of the network. **Middle:** > <former's decoded token distribution changes more gradually across middle layers. **Right:** > <former has lower entropy in early layers but declines less rapidly than the baseline in the final layers.

4.3 Predictive Dynamics via the Logit Lens

To understand how these geometric differences in the residual stream affect model predictions, we project intermediate hidden states into vocabulary space using the logit lens (nostalgebraist, 2020). Specifically, at each layer, we decode hidden states by applying the final RMS normalization followed by the unembedding matrix. As in §4.2, we treat > <former's effective wide residual stream as its hidden state, restricting to the residual dimensions visible to the unembedding.

For each layer, we measure the target-token log probability, the entropy of the decoded token distribution, and the layer-to-layer KL divergence between adjacent logit-lens distributions. We symmetrize this KL by averaging the two directions, using it as a proxy for how rapidly the decoded distribution changes with depth.

Figure 8 shows that > <former assigns higher probability to the target token, with lower decoded-distribution entropy, through much of the early-to-middle network. At the same time, its decoded token distribution changes more gradually across layers, as reflected by lower layer-to-layer KL. In the final layers, the distribution changes rapidly again as probability mass concentrates on the target token.

4.4 Ablations

We analyze alternative methods of expanding dimensions. Beyond our default method that carry forward features by copying coordinates through the residual stream, we also consider (1) padding with 0s and (2) training a projection layer to predict the extra dimensions from the previous layer representation.[10] For each, we also sweep over multiple hyperparameter configurations and report the best loss. We ablate at the 500M scale, and Table 4 shows that copying features performs the best.

Expansion Method	Loss
Constant-width	3.138
Carry-forward	3.099
Zero Padding	3.124
Projection	3.150

Table 4: Performance comparison of different methods to expand extra dimensions at 500M. **Simply carrying forward features from lower layers performs the best.**

5 Limitations

A major caveat is that our approach adds significant complexity for efficient training. Concretely, for efficient training one would need to develop and optimize kernels for many different shapes, each of which has different latency, memory footprint, and compute profiles. The fixed-residual construction also potentially adds overhead, since the slicing, copying, and zero-padding around a global residual stream wider than the baseline d introduce extra kernel launches, though much of this could be mitigated via kernel fusion. Heterogeneous per-layer widths are further in tension with standard tensor/pipeline parallelism techniques.

[10] We also tried training a projection to predict the entire new layer representation, not just the *extra* dimensions, but it is empirically unstable and diverges during training.

We stress, however, that these are implementation rather than algorithmic limitations: variable-width transformers are still matmul-rich, and the gap we describe reflects the fact that current infrastructure has been heavily optimized for the uniform-width regime rather than any intrinsic property of the architecture. We expect that purpose-built kernels would close much of the gap between theoretical and realized efficiency.

More broadly, while we are not calling for immediate adoption of > <formers, we hope that future architecture research can capitalize on this previously unnoticed degree of freedom in design.

6 Related Work

Nonuniform allocation of width in transformers. Several transformer variants allocate parameters nonuniformly across depth. DeLighT uses block-wise scaling, making earlier blocks shallower/narrower and later blocks deeper/wider (Mehta et al., 2020). OpenELM adopts layerwise scaling in decoder-only language models by varying attention and feed-forward dimensions across layers (Mehta et al., 2024). Recent layerwise-scaling variants explore framed, reverse, and crown allocation profiles (Baroian & Notebomer, 2025). Ikeda et al. (2025) study the layerwise importance of feed-forward networks by reallocating MLP capacity and find benefits from concentrating MLPs in middle layers. Our work differs from these approaches in varying the full block hidden dimension, rather than only the attention-head count, MLP multiplier, or lightweight block internals. This requires addressing how variable-width blocks interact with the residual stream; our fixed-residual construction lets inactive coordinates bypass narrower blocks.

Bottleneck across sequence length. There has also been work that performs compression across sequence length. Funnel-Transformer gradually shortens the sequence of hidden states and later recovers token-level representations for prediction (Dai et al., 2020). Hourglass Transformers downsample and upsample activations to build an explicit hierarchical language model (Nawrot et al., 2022). Perceiver models use cross-attention to distill high-dimensional inputs into a compact latent bottleneck before applying transformer-style processing (Jaegle et al., 2021). These methods primarily bottleneck the number of tokens or latent slots. Our architecture instead preserves the token sequence length and introduces a bottleneck in hidden width across depth.

Bottleneck designs outside Transformers. Bottleneck architectures have a long history outside of transformers. The U-Net and stacked hourglass networks use encoder–decoder structures that repeatedly reduce and recover spatial resolution, often with skip connections that preserve high-resolution information (Ronneberger et al., 2015; Newell et al., 2016). Other architectures introduce bottlenecks along the channel dimension: ResNets use bottleneck residual blocks to reduce the cost of deep convolutional networks (He et al., 2016), while MobileNetV2 uses inverted residual blocks with linear bottlenecks for efficient vision models (Sandler et al., 2018). However, transformer applications have mostly worked with non-bottleneck architectures in the channel dimension.

Hyper-Connections. By expanding residual-stream capacity, > <former is conceptually related to Hyper-Connections (HC) (Zhu et al., 2025; Xie et al., 2026; DeepSeek-AI, 2026). However, the mechanisms are different: HC uses learned mixing between multiple residual streams, whereas > <former uses deterministic slicing and carry-forward within a single global residual stream. In narrower layers, inactive coordinates bypass the block and are reintroduced when the width expands. Thus, > <former provides a complementary way to vary residual capacity without the learned residual-mixing matrices that Xie et al. (2026) identify as a source of large-scale HC instability.

7 Conclusion

In this work, we challenge the standard assumption of uniform capacity allocation across transformer depth by introducing the > <former, a variable-width architecture. Across evaluations from 200M to 3B parameters (dense and MoE), parameter-matched > <formers outperform uniform baselines, while mathematically and empirically reducing both FLOPs and KV cache memory. Furthermore, our analyses reveal that this bottleneck design may act as a structural regularizer, forcing the network to utilize its representation space more evenly. These findings demonstrate that nonuniform width allocation is an efficient and promising strategy for scaling future language models.

Acknowledgments

This study was supported in part by the MIT-IBM Watson AI Lab and the National Science Foundation under CAREER Award No. 2441872 and NSF grant No. CCF-21-12665.

References

Baroian, A. and Notebomer, K. Crown, frame, reverse: Layer-wise scaling variants for llm pre-training. *arXiv preprint arXiv:2509.06518*, 2025.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N. L., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. PaLM: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

Clark, D. G., Marschall, O., van Meegen, A., and Litwin-Kumar, A. Connectivity structure and dynamics of nonlinear recurrent neural networks. *Phys. Rev. X*, 15:041019, Nov 2025. doi: 10.1103/2jt7-c8cq. URL https://link.aps.org/doi/10.1103/2jt7-c8cq.

Dai, Z., Lai, G., Yang, Y., and Le, Q. V. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *Advances in Neural Information Processing Systems*, volume 33, pp. 4271–4282. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/2cd2915e69546904e4e5d4a2ac9e1652-Paper.pdf.

de Llano, E. Q., Arroyo, A., Barbero, F., Dong, X., Bronstein, M. M., LeCun, Y., and Shwartz-Ziv, R. Attention sinks and compression valleys in LLMs are two sides of the same coin. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=c5TFhCJ6fs.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446/.

Geva, M., Caciularu, A., Wang, K., and Goldberg, Y. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language*

Processing, pp. 30–45, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.3. URL https://aclanthology.org/2022.emnlp-main.3/.

Groeneveld, D., Beltagy, I., Walsh, E., Bhagia, A., Kinney, R., Tafjord, O., Jha, A., Ivison, H., Magnusson, I., Wang, Y., Arora, S., Atkinson, D., Authur, R., Chandu, K., Cohan, A., Dumas, J., Elazar, Y., Gu, Y., Hessel, J., Khot, T., Merrill, W., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M., Pyatkin, V., Ravichander, A., Schwenk, D., Shah, S., Smith, W., Strubell, E., Subramani, N., Wortsman, M., Dasigi, P., Lambert, N., Richardson, K., Zettlemoyer, L., Dodge, J., Lo, K., Soldaini, L., Smith, N., and Hajishirzi, H. OLMo: Accelerating the science of language models. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15789–15809, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.841. URL https://aclanthology.org/2024.acl-long.841/.

Gromov, A., Tirumala, K., Shapourian, H., Glorioso, P., and Roberts, D. The unreasonable ineffectiveness of the deeper layers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ngmEcEer8a.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hill, M. O. Diversity and evenness: A unifying notation and its consequences. *Ecology*, 54(2):427–432, 1973. doi: https://doi.org/10.2307/1934352. URL https://esajournals.onlinelibrary.wiley.com/doi/abs/10.2307/1934352.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J. W., and Sifre, L. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Ikeda, W., Yano, K., Takahashi, R., Lee, J., Shibata, K., and Suzuki, J. Layerwise importance analysis of feed-forward networks in transformer-based language models. *arXiv preprint arXiv:2508.17734*, 2025.

Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., and Carreira, J. Perceiver: General perception with iterative attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664. PMLR, 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Levine, Y., Wies, N., Sharir, O., Bata, H., and Shashua, A. Limits to depth efficiencies of self-attention. *Advances in Neural Information Processing Systems*, 33:22640–22651, 2020.

Li, J., Fang, A., Smyrnis, G., Ivgi, M., Jordan, M., Gadre, S. Y., Bansal, H., Guha, E. K., Keh, S., Arora, K., Garg, S., Xin, R., Muennighoff, N., Heckel, R., Mercat, J., Chen, M. F., Gururangan, S., Wortsman, M., Albalak, A., Bitton, Y., Nezhurina, M., Abbas, A. K. M., Hsieh, C.-Y., Ghosh, D., Gardner, J. P., Kilian, M., Zhang, H., Shao, R., Pratt, S. M., Sanyal, S., Ilharco, G., Daras, G., Marathe, K., Gokaslan, A., Zhang, J., Chandu, K., Nguyen, T., Vasiljevic, I., Kakade, S. M., Song, S., Sanghavi, S., Faghri, F., Oh, S., Zettlemoyer, L., Lo, K., El-Nouby, A., Pouransari, H., Toshev, A. T., Wang, S., Groeneveld, D., Soldaini, L., Koh, P. W., Jitsev, J., Kollar, T., Dimakis, A., Carmon, Y., Dave, A., Schmidt, L., and Shankar, V. Datacomp-LM: In search of the next generation of training sets for language models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=CNWdWn47IE.

Litwin-Kumar, A., Harris, K. D., Axel, R., Sompolinsky, H., and Abbott, L. Optimal degrees of synaptic connectivity. *Neuron*, 93(5):1153–1164.e7, 2017. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2017.01.030. URL https://www.sciencedirect.com/science/article/pii/S0896627317300545.

McLeish, S., Kirchenbauer, J., Miller, D. Y., Singh, S., Bhatele, A., Goldblum, M., Panda, A., and Goldstein, T. Gemstones: A model suite for multi-faceted scaling laws. *arXiv preprint arXiv:2502.06857*, 2025.

Mehta, S., Ghazvininejad, M., Iyer, S., Zettlemoyer, L., and Hajishirzi, H. Delight: Deep and light-weight transformer. *arXiv preprint arXiv:2008.00623*, 2020.

Mehta, S., Sekhavat, M. H., Cao, Q., Horton, M., Jin, Y., Sun, C., Mirzadeh, I., Najibi, M., Belenko, D., Zatloukal, P., et al. OpenELM: An efficient language model family with open training and inference framework. *arXiv preprint arXiv:2404.14619*, 2024.

Meng, K., Bau, D., Andonian, A. J., and Belinkov, Y. Locating and editing factual associations in GPT. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Nawrot, P., Tworkowski, S., Tyrolski, M., Kaiser, L., Wu, Y., Szegedy, C., and Michalewski, H. Hierarchical transformers are more efficient language models. In Carpuat, M., de Marneffe, M.-C., and Meza Ruiz, I. V. (eds.), *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 1559–1571, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.117. URL https://aclanthology.org/2022.findings-naacl.117/.

Newell, A., Yang, K., and Deng, J. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pp. 483–499. Springer, 2016.

nostalgebraist. Interpreting GPT: the logit lens. LessWrong, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Petty, J., Steenkiste, S., Dasgupta, I., Sha, F., Garrette, D., and Linzen, T. 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.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Roy, O. and Vetterli, M. The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference*, pp. 606–610, 2007.

Sajjad, H., Dalvi, F., Durrani, N., and Nakov, P. On the effect of dropping layers of pre-trained transformer models. *Comput. Speech Lang.*, 77(C), January 2023. ISSN 0885-2308. doi: 10.1016/j.csl.2022.101429. URL https://doi.org/10.1016/j.csl.2022.101429.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Shazeer, N. GLU variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Skean, O., Arefin, M. R., Zhao, D., Patel, N. N., Naghiyev, J., LeCun, Y., and Shwartz-Ziv, R. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=WGXb7UdvTX.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Tay, Y., Dehghani, M., Rao, J., Fedus, W., Abnar, S., Chung, H. W., Narang, S., Yogatama, D., Vaswani, A., and Metzler, D. Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*, 2021.

Team, K., Chen, G., Zhang, Y., Su, J., Xu, W., Pan, S., Wang, Y., Wang, Y., Chen, G., Yin, B., Chen, Y., Yan, J., Wei, M., Zhang, Y., Meng, F., Hong, C., Xie, X., Liu, S., Lu, E., Tai, Y., Chen, Y., Men, X., Guo, H., Charles, Y., Lu, H., Sui, L., Zhu, J., Zhou, Z., He, W., Huang, W., Xu, X., Wang, Y., Lai, G., Du, Y., Wu, Y., Yang, Z., and Zhou, X. Attention residuals, 2026. URL https://arxiv.org/abs/2603.15031.

Tenney, I., Das, D., and Pavlick, E. BERT rediscovers the classical NLP pipeline. In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4593–4601, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1452. URL https://aclanthology.org/P19-1452/.

Xie, Z., Wei, Y., Cao, H., Zhao, C., Deng, C., Li, J., Dai, D., Gao, H., Chang, J., Yu, K., Zhao, L., Zhou, S., Xu, Z., Zhang, Z., Zeng, W., Hu, S., Wang, Y., Yuan, J., Wang, L., and Liang, W. mhc: Manifold-constrained hyper-connections, 2026. URL https://arxiv.org/abs/2512.24880.

Yang, G., Yu, D., Zhu, C., and Hayou, S. Tensor programs VI: Feature learning in infinite depth neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=17pVDnpwwl.

Zhu, D., Huang, H., Huang, Z., Zeng, Y., Mao, Y., Wu, B., Min, Q., and Zhou, X. Hyper-connections. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9FqARW7dwB.

A Parameter-Matched Width Calculation

Here we derive how we instantiate a geometric width schedule from a bottleneck layer ℓ^* and bottleneck dimension d_{ℓ^*}, while matching the parameter count of a constant-width baseline. The derivation is in continuous width space; integer rounding is applied only after the parameter-matched widths have been determined.

Let L be the number of transformer layers, d the hidden dimension of the constant-width baseline, and v the vocabulary size. We assume the input and output embeddings maintain the baseline width d. The residual stream has a layer-dependent width d_ℓ, and resizing between adjacent layers is parameter-free. We impose symmetric endpoint widths, $d_1 = d_L = \bar{d}$.

The layer widths follow a geometric progression on each side of the bottleneck:

$$d_\ell = \begin{cases} \alpha^- d_{\ell-1}, & 1 < \ell \le \ell^*, \\ \alpha^+ d_{\ell-1}, & \ell^* < \ell \le L, \end{cases}$$

where $\alpha^- \le 1$ and $\alpha^+ \ge 1$. Symmetric endpoints imply $(\alpha^-)^{\ell^*-1}(\alpha^+)^{L-\ell^*} = 1$, which constrains α^+ for any candidate $\alpha^- \in (0, 1]$:

$$\alpha^+ = (\alpha^-)^{-\frac{\ell^*-1}{L-\ell^*}}.$$

Thus, the shape is strictly determined by α^-. We define the dimensionless factors $c_\ell(\alpha^-)$ such that $d_\ell = \bar{d}\, c_\ell(\alpha^-)$:

$$c_\ell(\alpha^-) = \begin{cases} (\alpha^-)^{\ell-1}, & 1 \le \ell \le \ell^*, \\ (\alpha^-)^{\ell^*-1}(\alpha^+)^{\ell-\ell^*}, & \ell^* < \ell \le L. \end{cases}$$

To match the parameter count of the constant-width baseline, we must account for the dominant layer parameters and endpoint corrections. For a dense transformer block with SwiGLU, the per-layer parameter count scales with $K d_\ell^2$, where $K = 4 + N_m E$ ($E = 4$ is the MLP expansion factor, and

$N_m = 3$ for SwiGLU is the number of MLP projections). Ignoring layer norm and bias terms, the baseline parameter count is $P_{\text{base}} = 2vd + LKd^2$.

Because the embeddings are fixed at width d, if our schedule requires $\bar{d} > d$ (as is the case for ><former), we pad the initial embeddings with 0s and truncate the final unembeddings. This results in unused parameters in the first attention layer and the final MLP layer. Specifically, the first attention QKV map and the last MLP output map contain $3\bar{d}(\bar{d} - d)$ and $E\bar{d}(\bar{d} - d)$ unused parameters, respectively. The total endpoint correction is therefore:

$$W_{\text{end}}(\bar{d}) = \mathbf{1}\{\bar{d} > d\}\,(3 + E)\bar{d}(\bar{d} - d).$$

Equating the valid parameters of our variable-width model to the baseline gives:

$$K\bar{d}^2 S_2(\alpha^-) - W_{\text{end}}(\bar{d}) = LKd^2,$$

where $S_2(\alpha^-) = \sum_{\ell=1}^{L} c_\ell (\alpha^-)^2$.

Substituting in $W_{\text{end}}(\bar{d})$, we can simplify this to:

$$\Big[KS_2(\alpha^-) - \mathbf{1}\{\bar{d} > d\}\,(3 + E)\Big]\bar{d}^2 + \Big[\mathbf{1}\{\bar{d} > d\}\,d(3 + E)\Big]\bar{d} - LKd^2 = 0.$$

Because the coefficients depend on the piecewise indicator $\mathbf{1}\{\bar{d} > d\}$, we solve this by assuming a state for the indicator (either 0 or 1), applying the standard quadratic formula to find the positive root, and selecting the root that is self-consistent with our assumption. This expresses the valid endpoint width as a function of α^-, which we denote as $\bar{d}_{\alpha^-}$. The bottleneck width for the given α^- is then:

$$b(\alpha^-) = \bar{d}_{\alpha^-}(\alpha^-)^{\ell^* - 1}.$$

We use a 1D numerical solver over $\alpha^- \in (0, 1]$ to solve for $b(\alpha^-) = d_{\ell^*}$, where d_{ℓ^*} is the desired bottleneck dimension. Finally, the continuous widths are rounded to the nearest multiple of the attention head dimension Q to ensure compatibility:

$$\widehat{d}_\ell = \text{round}\left(\frac{d_\ell}{Q}\right) Q.$$

B Dataset Statistics

In this section, we report the statistics of the datasets we used in our evaluation in Table 3.

Table 5: Task evaluation configurations and metrics.

Task	Domain	Split / Instances	Metric
OpenBookQA	science QA	500	`acc_norm`
PIQA	physical commonsense	1,838	`acc_norm`
SciQ	science QA	1,000	`acc_norm`
ARC-Easy	grade-school science QA	2,376	`acc_norm`
ARC-Challenge	difficult science QA	1,172	`acc_norm`
BoolQ	yes/no reading comprehension	3,270	`acc`
COPA	causal commonsense	100	`acc`
HellaSwag	commonsense completion	10,042	`acc_norm`
WinoGrande	pronoun/coreference reasoning	1,267	`acc`
RACE	reading comprehension	1,045	`acc`
WikiText	language modeling	62 documents	perplexity
LAMBADA OpenAI	long-context word prediction	5,153	`acc`, perplexity

C Additional Results

The analysis in §4.1 shows that > <former achieves better activation density, but it does not account for activation magnitude. Large language models frequently develop severe outlier dimensions, rendering the remaining active dimensions computationally insignificant. To evaluate this, we compute the energy Participation Ratio (PR) over the MLP activations (Litwin-Kumar et al., 2017; Clark et al., 2025). Let $a_{t,i}$ denote the activation of dimension i for token t and $e_i = \sum_t a_{t,i}^2$ denote the total energy of dimension i across all tokens t. The effective number of utilized dimensions is given by $N_{\text{eff}} = (\sum_i e_i)^2 / \sum_i e_i^2$. Intuitively, N_{eff} acts as a continuous measure of representational equality: if a single outlier dimension hoards all the numerical energy, N_{eff} collapses to 1, regardless of the actual width d_ℓ. Conversely, if computational energy is distributed uniformly across all coordinates, N_{eff} reaches its theoretical maximum of d_ℓ. Thus, computing the width-normalized fraction (N_{eff}/d_ℓ) provides a measure of the fraction of effectively utilized dimensions. We report both the absolute and normalized PR in Figure 9.

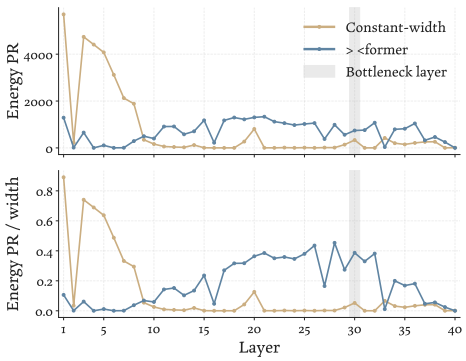

Figure 9: The Participation Ratio (PR; §4.1) of MLP activations in the 2B > <former vs. the 2B constant-width transformer. We show both the raw PR and the normalized PR by the layer width. **> <former has a higher PR in the middle layers, corresponding to more even usage of the activation dimensions in those layers.**

The results reveal a distinction between > <former and the constant-width transformer. For the baseline, its width-normalized energy utilization collapses to near zero ($< 5\%$) by around layer 10. In contrast, > <former more evenly distributes energy utilization in the middle layers by maintaining an absolute PR of roughly 1,000 effective dimensions, yielding a richer representation manifold. By restricting parameter availability, the bottleneck in > <former may act as a structural regularizer that encourages the network to pack a denser representation into the available capacity.