VARIABLE-WIDTH TRANSFORMERS

Anonymous authors
Paper under double-blind review

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 layers, allocating a fixed parameter and computation budget despite their potentially different computational roles. We empirically investigate nonuniform capacity allocation by proposing a $\times$-shaped ><former architecture. It maintains wider early and late layers while narrowing the middle layers, using 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). Consequently, dense ><formers reach the uniform baseline's final smoothed training loss with 15.6–23.1% fewer profiled GPU-hours, and generation throughput improves by 8.8% in one profiled setting. In analysis, we find qualitatively different residual-stream representations. Overall, 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 LM loss and reduces 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 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

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

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.

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: our models require fewer training and inference FLOPs than constant-width transformers, while also reducing 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, at the same parameter budget and training duration, > <formers achieve approximately a 3% relative improvement in perplexity over parameter-matched constant-width baselines, with about 10% smaller KV caches and 3% fewer FLOPs. These benefits also extend to mixture-of-experts transformers. Hardware profiling shows 15.6–23.1% fewer GPU-hours to reach the uniform baseline's final smoothed training loss across the dense models. We also analyze how optimal bottleneck width and location depend on model budget, providing empirical guidance for scaling nonuniform-width transformers. Finally, we examine MLP activations, residual-stream geometry, and predictive dynamics, finding qualitatively different representations from the constant-width baseline.

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.

We question why d must be constant. Prior work shows that transformer LM layers perform distinct functions that 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 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 parameter-matched, it requires fewer overall FLOPs and has a lower average layer width (and thus, lower KV cache size and lower I/O cost for moving activations). The dominant linear projection matrices (QKV, output, and MLP), contribute approximately $P_\ell \approx K d_\ell^2$ parameters per layer, where K is a constant depending on the number of projection matrices and the MLP expansion factor. Matching a constant-width baseline:

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

Applying the mean-square inequality, the average layer size is strictly smaller with non-constant d_ℓ:

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

For FLOPs, first, linear projection FLOPs per token is proportional to the weight count and remain identical to a parameter-matched 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 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

We first establish that a $\times$-shaped model works best, and then identify a parameterization of the 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. > <formers consistently achieve better loss and downstream task performance with fewer pre-training FLOPs 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 (Appendix Table 3). 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 achieves lower loss than the constant-width baseline despite this.

We pre-train on DCLM (Li et al., 2024). We train each model to 2.5× Chinchilla-optimal (Hoffmann et al., 2022), i.e., the number of trained tokens equals to 50 times the parameter count, e.g., 100B

[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)).

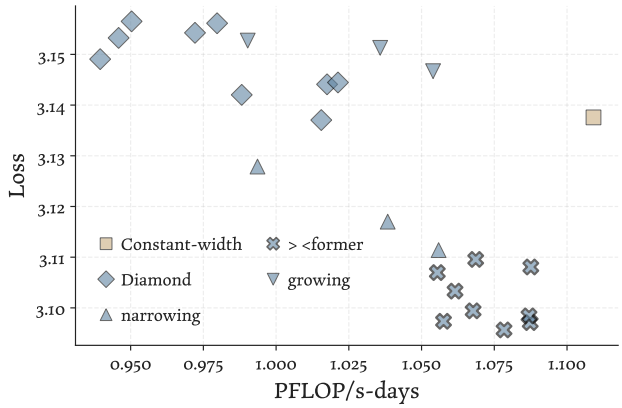


Figure 2: Comparing variable-width transformers with different shapes, each sweeping over multiple hyperparameter choices. **The $\times$-shaped model performs the best at 500M.**

tokens for the 2B model. The 3B/1B MoE model trains on 100B tokens too. We use length-4096 sequences, model-size-dependent batch sizes (Appendix Table 3), and the `cl100k_base`[7] tokenizer.

All models are trained with μP parameterization (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 linearly warm up the learning rate for the first 8% of steps and then decay it with a power schedule. We train in bfloat16 and omit bias 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 online 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 also compare average layer size, which is proportional to the KV cache size during inference.

3.2 The $\times$ Shape Works Best

At 500M, we compare the constant-width transformer with schedules of a $\diamond$ shape, a $\times$ shape, a $\vee$ shape, and a $\wedge$ shape. We sweep 3 values per relevant hyperparameter (ℓ^* and d_{ℓ^*}; §2): 9 runs for $\times$ and $\diamond$, and 3 runs for $\vee$ and $\wedge$, for which ℓ^* is irrelevant. In Figure 2, we plot their loss and the pre-training FLOPs. We see that the $\times$ shape consistently performs the best.

3.3 Finding a Specific Width Schedule

To choose the bottleneck layer index ℓ^* and dimension d_{ℓ^*} (§2) without a separate sweep at every model size, we parameterize them as ratios of total depth L and hidden size d: $\ell^* = r_\ell L$ and $d_{\ell^*} = r_d d$. 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 no 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 1 shows, at every tested scale, > <former outperforms the constant-width transformer with fewer FLOPs and smaller average layer width (and thus a smaller KV cache).

Scaling Laws. Figure 4 shows tight power-law fits for > <formers and constant-width transformers on loss vs. pre-training FLOPs (left) and average layer size (right) (Kaplan et al., 2020). From these scaling law curves, we estimate that > <former can achieve the 2B constant-width transformer's loss

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

[8] Since we do not repeat data, this equals held-out loss in expectation; separate held-out results are in §3.4.

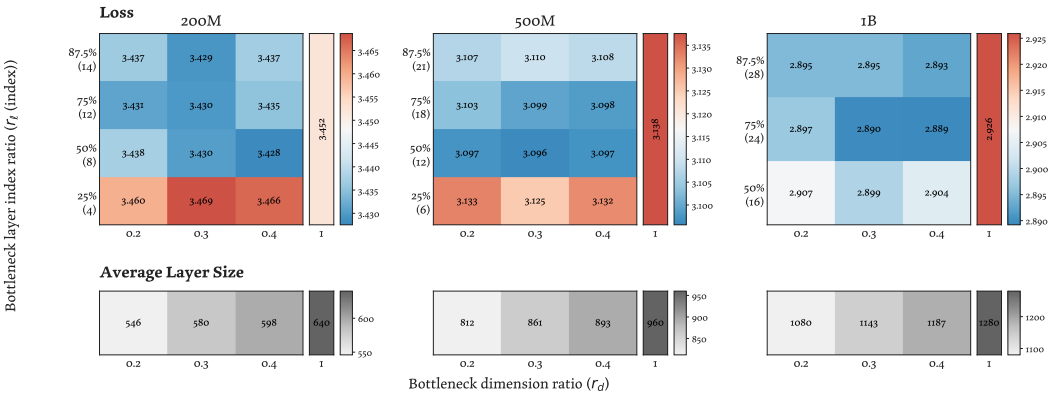


Figure 3: LM loss and average layer size by bottleneck layer index and dimension, as ratios of depth and base width: $\ell^* = r_\ell L$ and $d_{\ell *} = r_d d$; the bottom layer size row uses $r_\ell = 0.5$. The baseline uses $r_d = 1$. **This parameterization yields a relatively consistent performance trends across sizes.**

Table 1: Language modeling loss, pre-training FLOPs, average layer size, and GPU-hours to reach the constant-width baseline's final smoothed training loss. **> <former consistently achieves lower loss with lower pre-training FLOPs and average layer size.**

Size	Model	Loss	PFLOP/s-days	Avg layer size	GPU-hours to baseline loss
200M	Transformer	3.452	0.18	640	14.9
	> <former	**3.430**	**0.17 (−3.2%)**	**576 (−10.0%)**	**12.6 (−15.7%)**
500M	Transformer	3.138	1.11	960	67.5
	> <former	**3.099**	**1.07 (−3.7%)**	**855 (−11.0%)**	**52.7 (−22.0%)**
1B	Transformer	2.926	4.52	1,280	253
	> <former	**2.890**	**4.41 (−2.6%)**	**1,145 (−10.5%)**	**194 (−23.1%)**
2B	Transformer	2.751	16.92	1,600	1,184
	> <former	**2.726**	**16.49 (−2.5%)**	**1,426 (−10.9%)**	**999 (−15.6%)**
3B/1B MoE	Transformer	2.726	10.13	1,600	**1,845**
	> <former	**2.710**	**9.66 (−4.6%)**	**1,426 (−10.9%)**	1,895 (+2.7%)

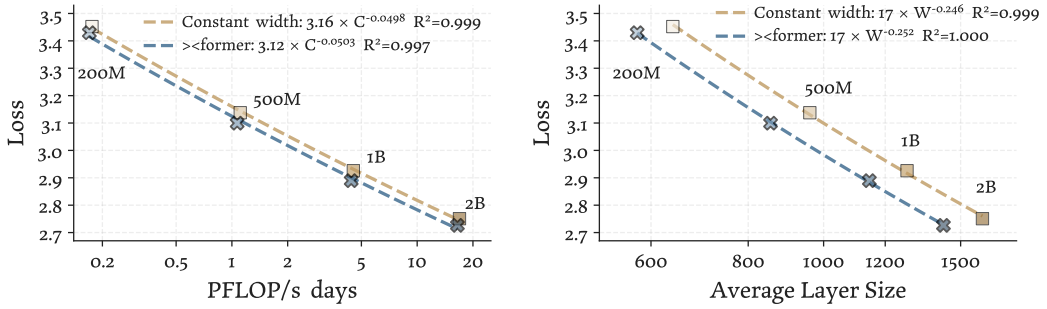


Figure 4: Language modeling loss vs. pre-training FLOPs (left) and average layer size (right). **> <former produces lower loss at smaller FLOP and average layer size costs.**

(2.751) with 77.8% of the FLOPs and 85.1% of the average layer width. As a check, refitting the FLOP curves using only 200M, 500M, and 1B correctly predicts the ranking of the withheld 2B models, with relative loss-prediction errors of 0.61% for the constant-width model and 1.31% for > <former. Furthermore, both fits have slightly steeper exponents for > <former, consistent with a growing relative loss advantage at fixed resources; whether this trend indeed persists at larger scales remains open.

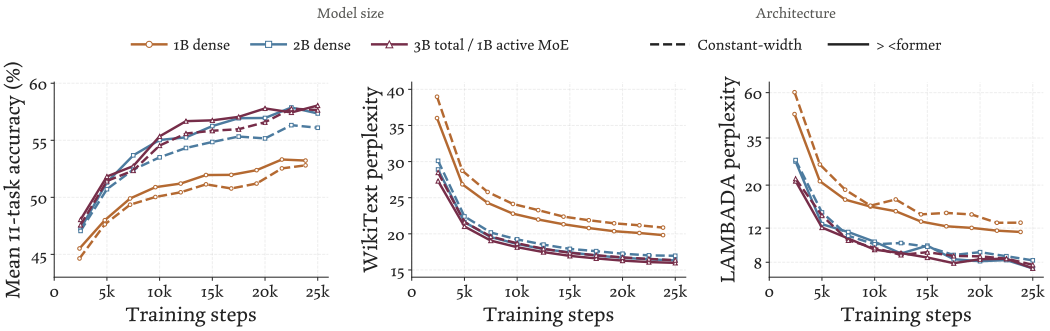


Figure 5: Downstream performance over 10 matched checkpoints per scale. Mean accuracy covers 10 multiple-choice tasks plus LAMBADA last-word accuracy. **> <former generally achieves higher mean accuracy and lower held-out perplexity evaluation.**

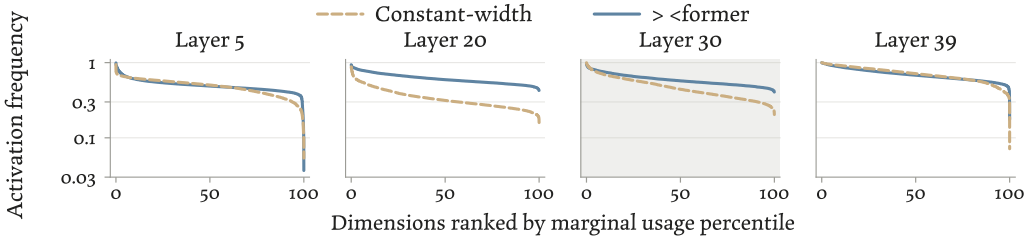


Figure 6: The utilization frequency of MLP activation dimensions in the 2B > <former vs. the 2B constant-width transformer, visualized separately for each layer. The shaded panel corresponds to the bottleneck layer. **> <former more evenly utilizes MLP activation dimensions.**

Downstream Tasks and Held-out Language Modeling Eval. We evaluate standard LM downstream benchmarks zero-shot using `lm-evaluation-harness` (Gao et al., 2024). This suite covers natural language understanding (NLU) tasks such as common-sense reasoning and reading comprehension, as well as perplexity tasks. For multiple-choice tasks, we report accuracy normalized for answer length when available, and otherwise standard accuracy. Dataset statistics and metrics are in Appendix Table 4. The gains are consistent throughout training: across 10 matched checkpoints for each of the 1B, 2B, and MoE model pairs, > <former achieves lower WikiText perplexity at all 30 checkpoints, higher mean 11-task accuracy at 29, and lower LAMBADA perplexity at 25 (Figure 5). At the final checkpoint, the 2B > <former leads on 8 of 10 multiple-choice tasks and the MoE > <former on 6 of 10 (Appendix Table 5). We again note that > <former achieves this with fewer FLOPs and memory, and also fewer active parameters for the MoE model (§3.1).

Hardware Efficiency As shown in Table 1, > <former reaches the baseline's final smoothed training loss in 15.6–23.1% fewer GPU-hours across all four dense scales. For the MoE, this slightly increases by 2.7%, though our implementation does not use kernels optimized for variable layer widths, which may contribute to this overhead. At 2B, reaching the baseline's final 10-task multiple-choice accuracy requires approximately 749 versus 1,066 profiled GPU-hours (-29.7%). At 2B, measured autoregressive generation throughput improves from 1,179 to 1,282 tokens/second ($+8.8\%$) at context length 1,024 and batch size 16. Peak per-GPU training memory falls from 49.65 to 46.97 GiB at 1B (16 GPUs), and from 26.15 to 24.78 GiB at 2B (64 GPUs).

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 uses a different representation strategy, with more even activation usage and higher residual matrix entropy in middle layers.

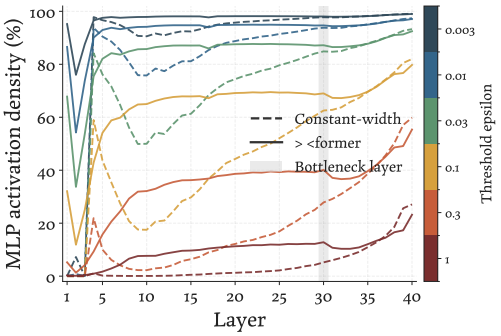


Figure 7: The density of MLP activations in the 2B > <former vs. the 2B constant-width transformer, across thresholds. **> <former consistently more densely activates MLP activation dimensions.**

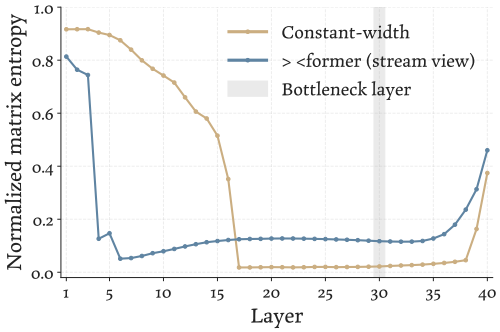


Figure 8: Normalized matrix entropy (§4.2) of the full residual stream in the 2B models. **> <former has higher middle-to-final layer entropy, which corresponds to more even usage of the residual dimensions in those layers.**

4.1 > <FORMER IMPROVES MLP ACTIVATION UTILIZATION

Intuitively, > <former enforces an information bottleneck that may encourage more effective use of its representation capacity. We operationalize this by inspecting intermediate MLP activation utilization. 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 exceeds a threshold. In Figure 7, we show that > <former has denser activations within MLPs across thresholds. At each of six thresholds from 0.003 to 1.0, its activation density is higher in 33–35 of the 40 layers.

Dense activations are not necessarily desirable, so we also measure each dimension's marginal utilization: how often it activates across tokens (threshold 0.1). In the interpretability literature, low marginal utilization and "dead" dimensions indicate under-utilized capacity (Bricken et al., 2023; Gao et al., 2025). Figure 6 shows four layers; Appendix Figure 10 shows all 40. While neither model has a perfectly even distribution, > <former has consistently better load-balancing between the activation dimensions. §D also shows a similar trend when also taking activation magnitude into account, where > <former likewise more evenly uses the activation dimensions in the middle layers of the network.

4.2 > <FORMER MAINTAINS HIGHER MIDDLE-LAYER RESIDUAL ENTROPY

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 singular values of the input-feature representation matrix $\mathbf{X}$ with rank r, computed on the WikiText-2 validation split. Closely related to the effective dimension metric (Hill, 1973; Roy & Vetterli, 2007), higher matrix entropy indicates that variance spreads more evenly across directions.

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 state, including coordinates that bypass narrower layers.

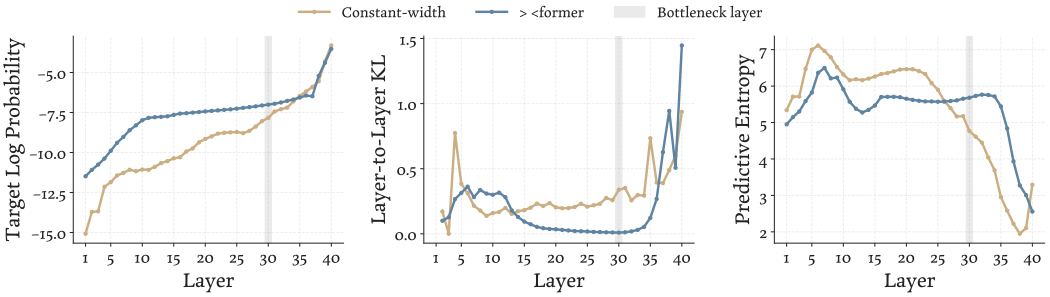


Figure 9: Logit lens analysis of the 2B > <former versus the constant-width baseline. **Left:** > <former assigns higher target-token probability through much of the network. **Middle:** > <former's decoded token distribution changes more gradually across middle layers. **Right:** > <former has lower entropy in early layers but declines less rapidly than the baseline in the final layers.

In Figure 8, the baseline exhibits a compression valley: in middle layers, its normalized entropy drops to near-zero, indicating that most variance is concentrated in a few directions despite the large width. This is consistent with prior findings (Skean et al., 2025; de Llano et al., 2026). In contrast, > <former has lower entropy in the early layers, before the width reduction, but avoids the middle-layer collapse. Throughout the bottleneck and final layers, > <former maintains higher normalized entropy.

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 average both KL directions as a proxy for how rapidly the decoded distribution changes with depth.

Figure 9 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, it 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 carries forward features by copying coordinates through the residual stream, we consider (1) padding with 0s and (2) training a projection layer to predict the extra dimensions from the previous layer representation.[9] For each, we also sweep over multiple hyperparameter configurations and report the best loss. At 500M, Table 2 shows that copying features performs best.

Table 2: Expansion methods at 500M. **Carrying features forward performs best.**

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

5 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.,

[9] 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.

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.

6 Conclusion and Limitations

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 reducing both FLOPs and KV-cache memory. Aggregate downstream accuracy is also higher at most checkpoints for the 1B, 2B, and MoE models. Furthermore, at 2B, our analyses show more even use of MLP activation dimensions and higher spectral entropy in the full residual stream. These observations are consistent with the hypothesis that the bottleneck acts as a structural regularizer, encouraging more even use of representation capacity. Together, these findings suggest that nonuniform width allocation is an efficient and promising strategy for scaling future language models.

A major caveat is that our approach adds complexity for efficient training, requiring optimized kernels for many different shapes with 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.

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 intrinsic to the architecture. We therefore expect purpose-built kernels to narrow 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 underexplored degree of freedom in design.

AI USE STATEMENT

Generative AI tools assisted with manuscript editing and organization and figure preparation. The authors are responsible for the final content.

REFERENCES

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

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L. Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

David G. Clark, Owen Marschall, Alexander van Meegen, and Ashok Litwin-Kumar. 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.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V. Le. 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.

Enrique Queipo de Llano, Alvaro Arroyo, Federico Barbero, Xiaowen Dong, Michael M. Bronstein, Yann LeCun, and Ravid Shwartz-Ziv. 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.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (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/.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (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/.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, William Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah Smith, and Hannaneh Hajishirzi. OLMo: Accelerating the science of language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (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/.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Dan Roberts. The unreasonable ineffectiveness of the deeper layers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ngmEcEer8a.

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.

M. O. Hill. 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.

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

Wataru Ikeda, Kazuki Yano, Ryosuke Takahashi, Jaesung Lee, Keigo Shibata, and Jun Suzuki. Layerwise importance analysis of feed-forward networks in transformer-based language models. *arXiv preprint arXiv:2508.17734*, 2025.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. 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.

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

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.

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

Ashok Litwin-Kumar, Kameron Decker Harris, Richard Axel, Haim Sompolinsky, and L.F. Abbott. 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.

Sean McLeish, John Kirchenbauer, David Yu Miller, Siddharth Singh, Abhinav Bhatele, Micah Goldblum, Ashwinee Panda, and Tom Goldstein. Gemstones: A model suite for multi-faceted scaling laws. *arXiv preprint arXiv:2502.06857*, 2025.

Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Delight: Deep and light-weight transformer. *arXiv preprint arXiv:2008.00623*, 2020.

Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, et al. OpenELM: An efficient language model family with open training and inference framework. *arXiv preprint arXiv:2404.14619*, 2024.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Lukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (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/.

Alejandro Newell, Kaiyu Yang, and Jia Deng. 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.

Jackson Petty, Sjoerd 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.

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

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

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 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.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

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

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. 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.

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

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, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, Yutian Chen, Junjie Yan, Ming Wei, Y. Zhang, Fanqing Meng, Chao Hong, Xiaotong Xie, Shaowei Liu, Enzhe Lu, Yunpeng Tai, Yanru Chen, Xin Men, Haiqing Guo, Y. Charles, Haoyu Lu, Lin Sui, Jinguo Zhu, Zaida Zhou, Weiran He, Weixiao Huang, Xinran Xu, Yuzhi Wang, Guokun Lai, Yulun Du, Yuxin Wu, Zhilin Yang, and Xinyu Zhou. Attention residuals, 2026. URL https://arxiv.org/abs/2603.15031.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In Anna Korhonen, David Traum, and Lluís Màrquez (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/.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mhc: Manifold-constrained hyper-connections, 2026. URL https://arxiv.org/abs/2512.24880.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. 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.

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

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. Layer ℓ actively processes d_ℓ residual coordinates,, 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 Kd_ℓ^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 hidden state to d coordinates before unembedding. 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 this parameter count for 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 $Q = 32$, ensuring an even dimension per attention head with 16 heads for RoPE:

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

Rounding can introduce a discrepancy relative to the continuous parameter count above.

B Model Hyperparameters

In this section, we report the model hyperparameters in Table 3.

Table 3: Pre-training hyperparameters. d is the hidden dimension of the constant-width baseline and the reference dimension for the variable-width schedules. Batch size is in the number of tokens.

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

Table 4: Task evaluation configurations and metrics.

Task	**Domain**	**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 Dataset Statistics

In this section, we report the statistics of the datasets we used in our evaluation in Table 4.

D Additional Results

Table 5: Model performance on standard LM evaluation datasets. **> <formers consistently outperform constant-width transformers on perplexity-based tasks, and the 2B > <former wins on most natural language understanding tasks.**

	Accuracy (↑)												Perplexity (↓)	
Model	ARC-C	ARC-E	BoolQ	COPA	HellaSwag	LAMBADA	OBQA	PIQA	RACE	SciQ	WinoGrande	Avg.	LAMBADA	WikiText
2B constant-width	33.3	59.4	59.2	75.0	55.9	55.2	33.4	73.7	34.6	79.6	57.7	56.1	8.18	16.96
2B > <former	34.5	63.3	60.8	74.0	57.8	56.1	34.0	74.4	33.8	82.1	60.1	57.4	7.44	16.32
MoE baseline	33.9	62.4	62.6	76.0	57.4	56.1	37.0	74.5	34.3	83.9	56.2	57.7	7.76	16.36
MoE > <former	33.6	61.0	59.5	81.0	58.7	56.1	37.6	75.0	35.0	80.2	60.8	58.0	7.44	15.98

Downstream task breakdown. Table 5 reports per-task zero-shot results for the 2B dense and MoE model pairs, supplementing the aggregated checkpoint-level comparison in §3.4.

Layerwise MLP activation utilization. Figure 10 extends the four-layer view in Figure 6 to all 40 layers of the 2B models. For each intermediate MLP dimension, we measure the fraction of WikiText-2 validation tokens for which its activation magnitude exceeds 0.1 (§4.1).

Energy participation ratio. The analysis in §4.1 shows that > <former has higher 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.

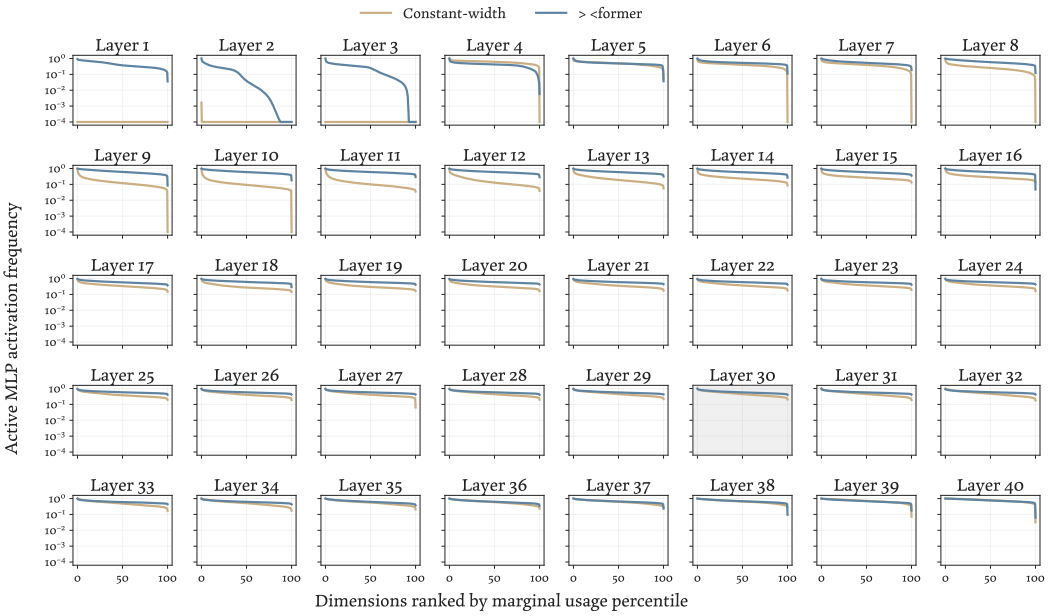


Figure 10: The utilization frequency of MLP activation dimensions in the 2B > <former vs. the 2B constant-width transformer, visualized separately for each layer. The shaded panel corresponds to the bottleneck layer. **> <former more evenly utilizes MLP activation dimensions.**

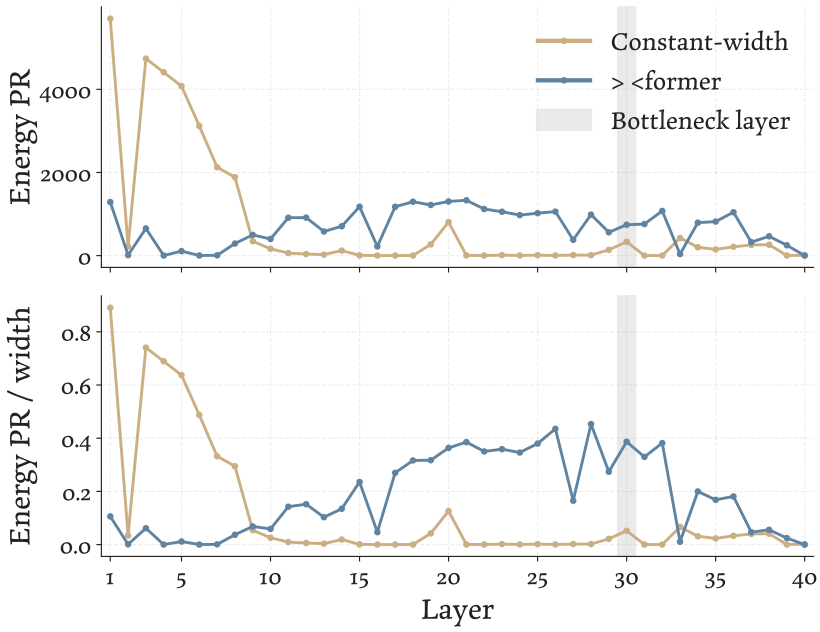


Figure 11: The Participation Ratio (PR; §4.1) of MLP activations in the 2B > <former vs. the 2B constant-width transformer. We show both the raw PR and the PR normalized by the intermediate MLP width. **> <former has a higher PR in the middle layers, corresponding to more even usage of the activation dimensions in those layers.**

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} measures how evenly activation energy is distributed: 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 $4d_\ell$. Thus, computing the width-normalized fraction ($N_{\text{eff}}/4d_\ell$) provides a measure of the fraction of effectively utilized dimensions. We report both the absolute and normalized PR in Figure 11.

The results reveal a distinction between > <former and the constant-width transformer. For the baseline, its width-normalized PR falls below 5% around layer 10 and stays below 5% in most subsequent layers. In contrast, > <former distributes squared activation magnitude more evenly in the middle layers, reaching an absolute PR of roughly 1,000 effective dimensions in many of them. 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.