TAPERED LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern language models, including transformer, recurrent, and memory-based variants, share a common chassis: a stack of identical layers in which parameters are allocated uniformly across depth. This default is inherited from the original transformer and has remained largely unchanged, yet a growing body of evidence suggests that layers contribute non-uniformly to the final output. We ask whether parameter capacity should reflect this asymmetry. In a controlled experiment at a fixed parameter budget, allocating more MLP capacity to earlier layers and less to later layers improves perplexity over a uniform-width baseline, while the reverse allocation, the direction taken by prior layer-wise scaling schemes, hurts. Building on this result, we propose *Tapered Language Models* (TLMs), in which MLP width decreases monotonically across depth under a fixed total budget. We focus on MLPs because they dominate parameter count across modern LM families and expose width as a single, clean axis of variation. Using a smooth cosine schedule selected once on a 440M Transformer and transferred unchanged, tapering improves average downstream accuracy and perplexity over matched uniform baselines in all six comparisons across three architectures (Transformer, Gated Attention, and Titans) at 760M and 1.3B parameters, with identical parameter count and FLOPs. These results suggest depth-aware capacity allocation as a simple, architecture-agnostic guideline for language model design.

1 INTRODUCTION

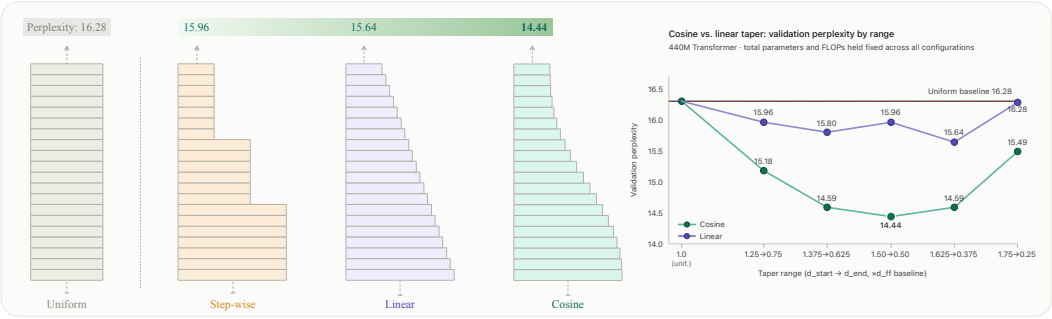


Figure 1: **Tapering MLP width improves perplexity at matched parameters and FLOPs.** *Left:* per-layer MLP intermediate width for a uniform baseline and three tapered schedules on a 440M Transformer; all configurations share the same total parameter count and FLOPs. Step-wise is the three-block wider-early allocation of Figure 2; linear and cosine are shown at their best taper range from Table 1. In-distribution validation perplexity is shown above each profile. The cosine taper reaches 14.44, compared with 16.28 for the uniform baseline. *Right:* validation perplexity as a function of the taper range $d_{\text{start}} \to d_{\text{end}}$ (in units of $d_{\text{ff}}^{\text{baseline}}$) for the cosine and linear schedules. Both trace a U-shape that bottoms out at an intermediate ratio; cosine is lower than linear at every range and is minimized at $1.50 \to 0.50$.

Modern language models, despite their surface diversity, share a common backbone. Transformers (Vaswani et al., 2017), gated-attention variants (Qiu et al., 2025), recurrent and state-space models (Beck et al., 2024; Orvieto et al., 2023; Peng et al., 2023; Sun et al., 2023; Hasani et al., 2022; Dao & Gu, 2024), and memory-based architectures (Schlag et al., 2021; Behrouz et al., 2024;

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.

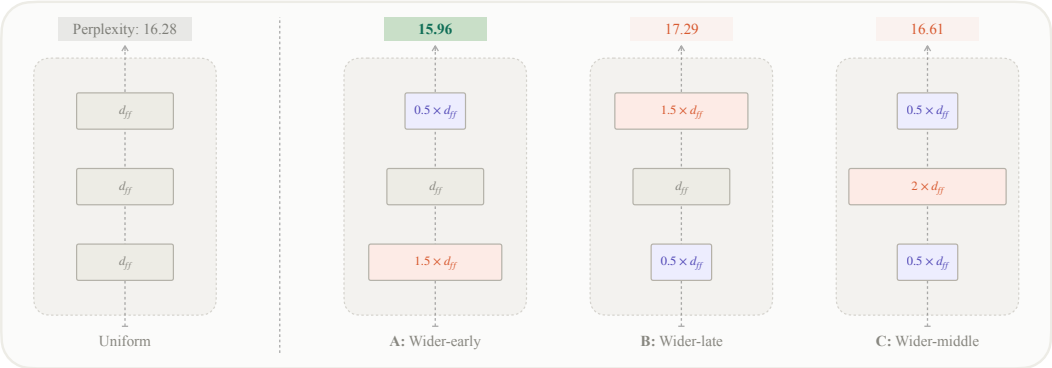


Figure 2: **Front-loading MLP capacity improves perplexity.** Layers of a 440M-parameter transformer are partitioned into three equal groups (early, middle, and late), and each group is assigned a different MLP intermediate width, with total parameter count held fixed across all configurations. Validation perplexity on a held-out split of FineWeb-Edu is reported above each configuration. Wider-early (A), which concentrates capacity in the early layers, achieves the lowest perplexity. The reverse allocation, wider-late (B), substantially hurts performance, and concentrating capacity in the middle layers (C) also degrades it. Under the same parameter budget, the direction of capacity allocation changes perplexity by more than a point.

2025c;b;a) differ in how they mix information across tokens, but they are all built from a stack of L identical layers, each combining a token-mixing module with a feed-forward network (FFN). Within this backbone, parameters are distributed uniformly across depth: every layer receives the same allocation, regardless of its position in the stack. As models have scaled by orders of magnitude, this uniformity has remained largely unexamined, a default carried through from the original transformer (Vaswani et al., 2017).

There is, however, growing evidence that layers contribute non-uniformly to the final output. Early-exit methods show that the residual stream often converges to its final prediction well before the last layer (Elbayad et al., 2020; Belrose et al., 2023), and layer-skipping frameworks demonstrate that later layers can be bypassed at inference time with minimal degradation in output quality (Elhoushi et al., 2024). Structured redundancy analyses find that many layers contribute negligibly to network function, and that deeper layers in particular can often be removed with surprisingly little performance loss (Men et al., 2025; Gromov et al., 2024; Lad et al., 2024). Interpretability work points in a similar direction: lower layers capture shallow syntactic patterns while upper layers encode more semantic ones (Geva et al., 2021), suggesting that both the importance and the nature of computation shift across depth. Most of this evidence is drawn from transformers, but the pattern across these independent lines of work is consistent: layer importance is non-uniform.

This raises a natural question: *if layer importance is non-uniform across depth, why is layer capacity uniform?* To test whether the direction of capacity allocation matters, we conduct a simple experiment on a 440M-parameter transformer. We partition the model's layers into three equal groups and assign different MLP intermediate widths to each, while holding the total parameter count fixed across configurations. We compare four allocations: a uniform baseline; wider-early, in which capacity is concentrated in the early layers and reduced in later ones; wider-late, the reverse; and wider-middle, which concentrates capacity in the middle layers (Figure 2).

Wider-early achieves the lowest perplexity at 15.96, compared with 16.28 for the uniform baseline. The reverse allocation, wider-late, raises perplexity by over a full point to 17.29, and concentrating capacity in the middle layers also degrades performance to 16.61. Under the same total parameter budget, the direction of allocation across depth changes perplexity by more than a point. This matters for existing practice: the wider-late direction is the one taken by the layer-wise scaling of OpenELM (Mehta et al., 2024), which increases FFN width from input to output layers, and prior studies of layer-wise allocation disagree on the preferred shape (§5). In our controlled setting, which varies only MLP width against an equal-cost baseline, front-loading capacity is the direction that helps.

The motivating experiment uses a coarse piecewise allocation with sharp transitions between groups. A natural follow-up is whether smoother allocations do better, and how to specify them. We propose *Tapered Language Models* (TLMs), in which MLP width decreases monotonically across depth under a fixed total budget, replacing sharp block transitions with a smooth decay from early to late layers. MLPs are the natural place to test this: they are the dominant parameter store in modern language models across architectural families, and their structure exposes a single width parameter, the intermediate dimension d_{ff}, that can be adjusted independently of the surrounding architecture. This choice is also consistent with interpretability work characterizing FFNs as key-value memories whose contents shift from shallow to semantic patterns with depth (Geva et al., 2021). We state the tapering condition generally in §2.2, since it may apply to any per-layer dimension that controls parameter count, but we test only MLP width in this work.

We select a cosine schedule on a 440M Transformer and carry it unchanged to three architectures with different token-mixing modules: standard Transformers (Vaswani et al., 2017), Gated Attention (Qiu et al., 2025), and Titans (Behrouz et al., 2024), at 760M and 1.3B parameters. Tapering improves average commonsense accuracy, WikiText perplexity, and LAMBADA perplexity in all six architecture/scale comparisons, with gains at both scales. To probe why, we measure how aligned each layer's update is with the residual stream of pretrained transformers, and find that MLP outputs become progressively more aligned with the existing residual through the second half of the network. This observation is consistent with tapering: capacity is reduced where layers appear to add the least new content.

Our contributions:

- We show in a controlled comparison at fixed budget that the direction of depth-wise MLP capacity allocation matters: front-loading capacity helps, while back-loading, the direction taken by prior layer-wise scaling, hurts. Building on this, we propose *Tapered Language Models* (TLMs), which taper MLP width monotonically across depth using smooth decay schedules (linear, cosine, and sigmoid) that preserve total parameter count and FLOPs.
- We show that a single cosine taper, selected on one 440M Transformer and transferred without re-tuning, improves average downstream accuracy over uniform baselines for three architectures (Transformer, Gated Attention, and Titans) at two scales, and report its practical throughput and memory cost.
- We provide a layer-wise analysis showing that MLP outputs become progressively more aligned with the residual stream through the second half of the network, an observation consistent with reducing MLP capacity in later layers.

2 Tapered Language Models

2.1 Background

We consider decoder-only language models built from a stack of L blocks, each composed of a token-mixing module $\mathcal{M}$ and an MLP $\mathcal{F}$ operating on a residual stream of dimension d. Given a hidden state $h_l \in \mathbb{R}^{N\times d}$ at layer l, the block computes

$$z_l = h_l + \mathcal{M}_l(h_l), \tag{1}$$

$$h_{l+1} = z_l + \mathcal{F}_l(z_l), \tag{2}$$

where layer normalization is omitted for brevity. Architectures differ in the choice of $\mathcal{M}$, from softmax attention (Vaswani et al., 2017) to gated attention (Qiu et al., 2025) to recurrent and memory-based mechanisms (Behrouz et al., 2024; 2025b), but $\mathcal{F}$ takes the same form across them: a projection from d to an intermediate dimension d_{ff}, a nonlinearity, and a projection back to d. Its parameter count is $2\,d\,d_{\text{ff}}$ for a standard FFN and $3\,d\,d_{\text{ff}}$ for gated variants such as SwiGLU (Shazeer, 2020), so both its parameters and its FLOPs are linear in d_{ff}. In standard practice, d_{ff} is a fixed multiple of d ($4d$, or $\frac{8}{3}d$ for gated variants) and is held constant across all L layers.

2.2 Tapering

Holding a per-layer dimension constant across depth is one point in a larger design space. Consider any component C whose dimension $d_C(l)$ at layer $l \in \{0, 1, \ldots, L-1\}$ controls its parameter count.

The uniform design fixes $d_C(l) = d_C^{\text{baseline}}$ for all l. We call a design *tapered* if instead

$$d_C(l+1) \le d_C(l) \quad \text{for } l = 0, \dots, L-2, \qquad \frac{1}{L}\sum_{l=0}^{L-1} d_C(l) = d_C^{\text{baseline}}, \tag{3}$$

so that capacity decreases (weakly) with depth while the component's total parameter budget is preserved. We test this condition on MLP width, which is shared across all architectures we study.

Tapering MLP width. We replace the constant d_{ff} with a per-layer width $d_{\text{ff}}(l)$ that decreases from d_{start} at the first layer to d_{end} at the last. Because each MLP's parameters and FLOPs are linear in its width, the budget condition in Equation 3 becomes

$$\frac{1}{L}\sum_{l=0}^{L-1} d_{\text{ff}}(l) = d_{\text{ff}}^{\text{baseline}}, \tag{4}$$

and satisfying it keeps total parameters, training FLOPs, and inference FLOPs identical to the uniform baseline; tapering only changes where they are spent. Realized throughput and peak memory can still differ, since layers are no longer identical in shape; we measure both in §3.4.

Decay schedules. The motivating experiment (Figure 2) used a discrete three-block allocation; here we use smooth decay. Writing $t = l/(L-1) \in [0, 1]$ for relative depth, we consider three schedules (Figure 3). *Linear* decays at a constant rate, with no plateau at either end:

$$d_{\text{ff}}(l) = d_{\text{start}} - (d_{\text{start}} - d_{\text{end}})\, t. \tag{5}$$

Cosine has soft plateaus at both ends and decays most steeply at the midpoint:

$$d_{\text{ff}}(l) = d_{\text{end}} + \frac{d_{\text{start}} - d_{\text{end}}}{2}\big(1 + \cos \pi t\big). \tag{6}$$

Sigmoid concentrates the transition in a narrow band around the midpoint, leaving most layers near d_{start} or d_{end}, with steepness set by k (we use $k = 10$):[1]

$$d_{\text{ff}}(l) = d_{\text{end}} + \frac{d_{\text{start}} - d_{\text{end}}}{1 + e^{k(t-0.5)}}. \tag{7}$$

All three schedules are symmetric about the midpoint, so each averages exactly to $(d_{\text{start}} + d_{\text{end}})/2$. Equation 4 therefore reduces to $d_{\text{start}} + d_{\text{end}} = 2\, d_{\text{ff}}^{\text{baseline}}$, and a configuration is fully specified by its schedule and its taper range, which we write in units of $d_{\text{ff}}^{\text{baseline}}$ (e.g., $1.5 \to 0.5$). The schedules differ only in how the transition is distributed across depth; we compare them empirically in §3.2.

Implementation details. Per-layer widths are rounded to the nearest multiple of 16 for efficient matrix operations on accelerator hardware. The first and last layer widths are fixed exactly to d_{start} and d_{end}, and interior widths are adjusted in 16-unit increments so that Equation 4 holds exactly and the widths remain monotonically decreasing. All other hyperparameters (residual dimension d, attention head count, key-value dimension) are identical to the uniform baseline.

3 Experiments

Our experiments follow a select-then-transfer protocol. We first select a tapering configuration (schedule and taper range) with a sweep on a single 440M Transformer++ (§3.2). We then fix that configuration and apply it, without re-tuning, to three architectures at two larger scales (§3.3). This design tests not only whether tapering helps, but whether a configuration chosen once transfers across architectures and scales.

[1] With $k = 10$ the sigmoid does not exactly reach d_{start} and d_{end} at the endpoints; the endpoint pinning described under *Implementation details* enforces them.

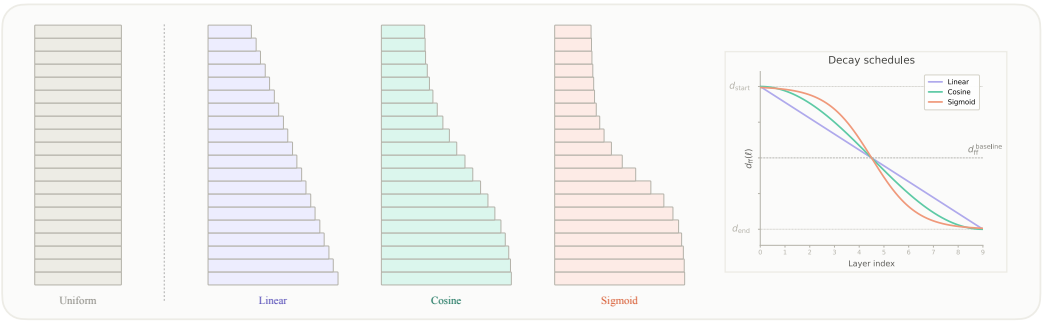


Figure 3: **Tapering smoothly reallocates MLP width across depth.** MLP intermediate dimension across layers for the uniform baseline and three tapered schedules, shown as per-layer bar widths (left) and as continuous decay curves (right) for a representative configuration. All configurations share the same total parameter count; the curves cross $d_{\text{ff}}^{\text{baseline}}$ at the midpoint because the per-layer widths average to $d_{\text{ff}}^{\text{baseline}}$ by construction.

3.1 Setup

Training. We follow the standard pre-training setup of recent studies: all models are trained on FineWeb-Edu (Penedo et al., 2024) using the Llama 2 tokenizer (Touvron et al., 2023) with a vocabulary size of 32K and a training sequence length of 4K tokens. The 440M, 760M, and 1.3B models are trained on 30B, 50B, and 100B tokens, respectively. We use AdamW with a cosine annealing schedule, a peak learning rate of 4×10^{-4}, a weight decay of 0.1, and a global batch size of 0.5M tokens. All hyperparameters follow the uniform-baseline setup and are identical between uniform and tapered models; the only difference is the per-layer MLP intermediate dimension.

Evaluation. For configuration selection (§3.2), we report in-distribution validation perplexity on a held-out split of FineWeb-Edu. For the transfer experiments (§3.3), we report out-of-distribution perplexity on WikiText (Merity et al., 2016) and LAMBADA (Paperno et al., 2016), and accuracy on eight commonsense reasoning benchmarks: LAMBADA, PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-easy and ARC-challenge (Clark et al., 2018), SIQA (Sap et al., 2019), and BoolQ (Clark et al., 2019).

3.2 Selecting a Tapering Configuration

We sweep the decay schedule (Equations 5–7) and the taper range on a 440M Transformer++, using five ranges, $\{1.25 \to 0.75,\ 1.375 \to 0.625,\ 1.5 \to 0.5,\ 1.625 \to 0.375,\ 1.75 \to 0.25\}$, for each of the three schedules. All 15 configurations match the uniform baseline in parameters and FLOPs. Table 1 reports validation perplexity for every cell.

Across all five taper ranges, cosine yields the lowest perplexity, linear the second-lowest, and sigmoid the highest. The gap between cosine and the other two dominates the table: even cosine's worst configuration ($1.75 \to 0.25$, 15.49) is lower than linear's best ($1.625 \to 0.375$, 15.64). This ordering is consistent with the schedule shapes described in §2: sigmoid's near-binary allocation behaves like a coarse two-block split, while cosine's soft endpoint plateaus and gradual mid-stack transition provide a smoother gradient of capacity across depth.

Holding the schedule fixed at cosine, perplexity follows a U-shape across the five ranges, reaching its minimum at $1.5 \to 0.5$, where tapering reduces perplexity from 16.28 to 14.44, a 1.84-point improvement at identical parameters and FLOPs. The gain does not depend on a precise choice of range: cosine improves over the uniform baseline at every range we tested, by 0.79 to 1.84 points. This robustness makes the configuration a practical default, and we fix cosine with $1.5 \to 0.5$ for all remaining experiments without further tuning. Narrower ranges are also a practical option: $1.25 \to 0.75$ still improves perplexity by 1.10 points with a smaller widest layer and lower peak memory (§3.4).

Table 1: **Selecting a tapering configuration.** In-distribution validation perplexity on the 440M Transformer++ across five taper ranges and three schedules. The Δ columns report the change relative to the uniform baseline (first row); green indicates improvement, red indicates regression, and saturation indicates magnitude. All 15 configurations match the uniform baseline in total parameters, training FLOPs, and inference FLOPs.

	Cosine		Linear		Sigmoid	
Taper range ($\times d_{\text{ff}}^{\text{baseline}}$)	ppl ↓	Δ	ppl ↓	Δ	ppl ↓	Δ
Uniform (1.00 → 1.00)			16.28			
1.25 → 0.75	15.18	−1.10	15.96	−0.32	16.44	+0.16
1.375 → 0.625	14.59	−1.69	15.80	−0.48	16.44	+0.16
1.50 → 0.50	**14.44**	**−1.84**	15.96	−0.32	16.12	−0.16
1.625 → 0.375	14.59	−1.69	15.64	−0.64	15.96	−0.32
1.75 → 0.25	15.49	−0.79	16.28	0.00	17.12	+0.84

3.3 Transfer Across Architectures and Scales

We now test whether this configuration, selected on a single 440M Transformer++, transfers to other architectures and scales. We apply it unchanged, with no per-architecture or per-scale tuning, to three architectures with different token-mixing modules: Transformer++, a Transformer (Vaswani et al., 2017) with Llama-style modifications; Gated Attention (Qiu et al., 2025), which adds output gating to softmax attention to remove attention sinks and improve sparsity; and Titans (Behrouz et al., 2024), which augments attention with a neural long-term memory module that learns to memorize at test time. We train each at 760M and 1.3B parameters, with and without tapering, at matched parameters and FLOPs. Table 2 reports the results.

Across all three architectures and both scales, the tapered model improves average commonsense accuracy over its uniform counterpart. WikiText and LAMBADA perplexity also improve in all six comparisons, and the improvements are broad rather than driven by a single task. These gains are more modest than those observed during selection (§3.2), but the two are not directly comparable: selection measures in-distribution validation perplexity on FineWeb-Edu, whereas Table 2 reports out-of-distribution perplexity and zero-shot accuracy. Moreover, neither the taper shape nor the taper range is re-tuned for these architectures or scales, so these results reflect what tapering delivers out of the box. Per-setting tuning may yield larger gains, but it requires a separate pretraining sweep for each architecture and scale, so we leave it to practitioners.

The transfer holds across token-mixing mechanisms ranging from softmax attention and gated attention to neural long-term memory. This suggests the benefit comes from how parameters are allocated across depth in the MLP stack, rather than from a property specific to one token-mixing module. It also holds at both scales, with average accuracy gains of up to 0.99 points.

3.4 Practical Cost

Tapering matches the uniform baseline exactly in total parameters and FLOPs; because layers differ in shape, we also measured realized training throughput and peak memory at four scales (Table 3). Throughput is unchanged up to 440M and within 6% of the uniform baseline at 760M and 1.3B. Peak memory is 8–17% higher from 440M upward, due to the widest layer (1.5× the baseline width). Both costs are controlled by the taper range: narrower ranges such as 1.25 → 0.75 reduce the widest layer while retaining a large part of the gain (§3.2).

Table 3: **Practical cost of tapering.** Training throughput and peak memory of tapered models relative to matched uniform baselines. Values above 1 in the memory row indicate higher memory use; values below 1 in the throughput row indicate lower throughput.

Tapered / uniform	124M	440M	760M	1.3B
Throughput	1.00	1.00	0.95	0.94
Memory	1.00	1.08	1.12	1.17

Table 2: **Transfer across architectures and scales.** Perplexity on WikiText and LAMBADA and accuracy on eight commonsense benchmarks for 760M (50B tokens) and 1.3B (100B tokens) models, comparing each backbone with and without tapering (shaded rows). All tapered models use the configuration selected in §3.2 (cosine, $1.5 \rightarrow 0.5$) without re-tuning. Avg. denotes the mean accuracy across reasoning tasks.

Model	Perplexity		Commonsense Reasoning								
	Wiki. ppl ↓	LMB. ppl ↓	LMB. acc ↑	PIQA acc ↑	Hella. acc_n ↑	Wino. acc ↑	ARC-e acc ↑	ARC-c acc_n ↑	SIQA acc ↑	BoolQ acc ↑	Avg. ↑
760M params / 50B tokens											
Transformer++	21.86	22.29	39.0	68.7	46.3	57.1	66.8	35.3	42.5	62.3	52.25
+ Tapered	21.42	21.25	40.1	69.3	47.0	57.3	66.7	35.9	43.0	63.4	**52.84**
Gated Attention	20.74	21.85	39.7	69.2	46.3	57.9	68.4	35.5	41.8	62.1	52.61
+ Tapered	19.98	21.44	40.0	69.3	46.8	57.8	69.1	35.8	41.6	62.6	**52.88**
Titans	21.58	23.09	39.2	67.7	50.0	52.8	68.0	35.6	41.4	63.7	52.30
+ Tapered	20.77	22.92	39.9	69.0	51.6	54.5	67.9	36.1	42.5	64.8	**53.29**
1.3B params / 100B tokens											
Transformer++	17.39	17.62	45.1	72.8	53.5	59.7	70.6	38.4	44.0	64.3	56.05
+ Tapered	17.17	16.93	45.7	72.5	53.4	60.0	70.8	38.3	44.6	65.7	**56.38**
Gated Attention	16.03	14.26	46.2	73.0	53.9	60.2	71.5	38.7	44.2	64.4	56.51
+ Tapered	15.92	14.11	46.5	73.6	53.8	60.4	71.7	38.7	44.5	65.2	**56.80**
Titans	16.05	14.19	46.9	73.1	53.5	59.9	72.4	39.3	43.9	64.8	56.73
+ Tapered	15.76	14.04	46.9	73.8	54.2	60.7	72.1	39.6	43.8	65.5	**57.08**

4 Layer-wise Novelty

What might explain the benefit of tapering? We measure how much novel content each layer writes into the residual stream of pretrained transformers with uniform capacity, and find that MLP outputs become progressively more aligned with the existing residual through the second half of the network. Later MLPs increasingly reinforce directions the residual already encodes rather than adding new ones, which is consistent with reducing their capacity.

We measure two cosine quantities per layer. Using the notation of §2.1 ($z_l = h_l + \mathcal{M}_l(h_l)$, $h_{l+1} = z_l + \mathcal{F}_l(z_l)$),

$$\rho_l^{\text{block}} = \cos\big(h_{l+1} - h_l,\, h_l\big), \tag{8}$$

$$\rho_l^{\text{MLP}} = \cos\big(\mathcal{F}_l(z_l),\, h_l\big), \tag{9}$$

averaged over tokens. The block-update quantity captures the layer's full additive contribution; the MLP-only quantity isolates the component being tapered.[2] $\rho \approx 0$ corresponds to writing content orthogonal to the residual; $\rho > 0$ corresponds to reinforcing a direction the residual already encodes. A rising trend with depth therefore signals diminishing novelty.

We compute ρ_l^{block} and ρ_l^{MLP} on 2048 tokens of WikiText-2 (Merity et al., 2016) across the GPT-2 (Radford et al., 2019) family, which provides publicly available pretrained checkpoints across a range of scales under a controlled architecture, allowing us to isolate the depth-wise pattern without confounds from training variation. The first and last layers are omitted from the correlation analysis as boundary cases. The pattern in Figure 4 is consistent across the family: both quantities climb through the second half of the network in every model. Pearson correlation with layer index is positive in every measurement and tighter for the MLP than for the full block: ρ_l^{MLP} correlations range from $r = 0.49$ to $r = 0.71$, and ρ_l^{block} from $r = 0.27$ to $r = 0.71$. The rise in ρ_l^{MLP} suggests

[2] $\mathcal{F}_l(z_l)$ is added to z_l in the residual stream rather than to h_l. We use h_l as the reference for both ρ_l^{block} and ρ_l^{MLP} so that the two quantities share a single reference, the residual stream entering the block.

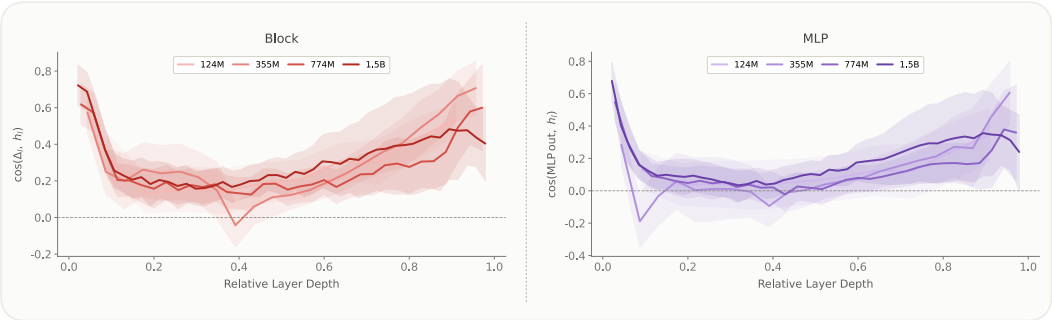


Figure 4: **Layer updates become more aligned with the residual stream in the second half of the network.** Cosine similarity between each layer's update and the residual stream entering its block, as a function of relative layer depth, across the GPT-2 family (124M to 1.5B; lightest to darkest shade). Left: full block update, ρ_l^{block}. Right: MLP-only, ρ_l^{MLP}. Both quantities rise through the second half of the network in every model size, with the MLP signal showing the more consistent rise. Shaded bands show ± 1 standard deviation across tokens; the dashed line marks $\rho = 0$.

that the MLP's own contribution becomes less novel with depth, not merely that attention dominates the update in later layers; the rise in ρ_l^{block} suggests that the pattern extends to the layer as a whole.

Connection to tapering. If later MLPs at uniform capacity mostly reinforce directions already present in the residual, part of their hidden dimension may be underused, and reducing it should cost little. This is consistent with the direction of tapering, which removes capacity from later layers, and with the asymmetry of Figure 2, where adding capacity to later layers hurts. We present this analysis as an account consistent with our results rather than a causal demonstration. In particular, it speaks to where capacity can be reduced, not to where the freed capacity is best placed, which we determine empirically.

5 Related Work

Language model families. Today's language models share the same per-block design despite differing in their token-mixing modules. Transformers (Vaswani et al., 2017) and gated-attention variants (Qiu et al., 2025) use softmax attention; linear recurrent and state-space models (Schlag et al., 2021; Tiezzi et al., 2024; Sun et al., 2023; Peng et al., 2023; 2024; 2025; Smith et al., 2023; Hasani et al., 2022; 2021; Dao & Gu, 2024; Beck et al., 2024; Orvieto et al., 2023; De et al., 2024; Liu et al., 2024; Siems et al., 2025; Ren et al., 2025; Merrill et al., 2026) replace it with data-dependent recurrences for sub-quadratic scaling; and memory-based architectures (Sun et al., 2024; Behrouz et al., 2024; 2025c;b;a; Wang et al., 2025; Hu et al., 2026; Li et al., 2026) augment it with learnable memory modules that adapt at test time. In all of these, each block pairs a token-mixing module with an MLP, and parameters are allocated uniformly across depth. Tapering targets this shared default.

Layer-wise capacity allocation. The work closest to ours redistributes parameters across layers at a fixed budget. DeLighT (Mehta et al., 2020) uses block-wise scaling that varies FFN multipliers and head counts jointly across depth, and OpenELM (Mehta et al., 2024) uses layer-wise scaling that increases both FFN width and attention heads from input to output layers, placing more capacity in later layers. Baroian & Notebomer (2025) compare several layer-wise scaling shapes at fixed budget without identifying a clear winner, and Ikeda et al. (2025) deactivate FFNs in some layers entirely, concentrating capacity in a contiguous band. Together, these works indicate that uniform allocation is suboptimal but disagree on the right shape. Our setup is designed to resolve this: it keeps every layer active, varies a single axis (MLP width) rather than several jointly, and compares against a uniform baseline with identical parameters and FLOPs. Under these controls, front-loading capacity helps in the motivating experiment (Figure 2) and in all three architectures we evaluate (§3.3).

Post-hoc heterogeneous capacity. A separate line of work produces models with non-uniform per-layer capacity after training. Selective rank reduction (Sharma et al., 2023) and structured

pruning (Ashkboos et al., 2024; Ling et al., 2024) remove MLP capacity layer by layer, and Puzzle (Bercovich et al., 2024) uses distillation-based architecture search to select layer-specific attention and FFN variants, including reduced FFN widths. These methods start from a trained, typically larger model and require additional pruning, fine-tuning, or distillation stages. Tapering instead fixes a non-uniform allocation before pretraining, at an unchanged parameter and compute budget and with no teacher model. The two approaches are complementary: pruning or distillation can be applied to a tapered model just as to a uniform one.

Other forms of depth-wise variation. Several methods vary computation across depth along axes other than static MLP width: pooling sequence length (Dai et al., 2020), routing compute dynamically to tokens (Raposo et al., 2024; Bae et al., 2026), dropping layers during training (Fan et al., 2019), and nesting variable-size blocks within layers for elastic inference (Kudugunta et al., 2024). Mixture-of-Experts (Fedus et al., 2022) varies capacity across tokens rather than layers. These directions address different questions from ours and could in principle be combined with tapering.

Layer importance. As discussed in §1, early-exit, layer-skipping, and redundancy analyses show that later layers often contribute less to the final output (Elbayad et al., 2020; Schuster et al., 2022; Elhoushi et al., 2024; Belrose et al., 2023; Men et al., 2025; Gromov et al., 2024; Lad et al., 2024), and interpretability work finds that the nature of computation shifts across depth (Geva et al., 2021; Voita et al., 2019; Skean et al., 2025; Bayat et al., 2025). These works motivate tapering; our analysis in §4 complements them with a direct measurement of how the novelty of MLP updates changes with depth, and our experiments turn this observation into a pretraining-time allocation rule.

6 Limitations

Our schedule and width-ratio sweep is conducted only on the 440M Transformer. We then transfer the selected cosine schedule with taper range $1.5 \rightarrow 0.5$ unchanged to the larger models and alternative architectures. This design provides a deliberately strict test of transferability: the improvements in the main experiments show that the selected configuration is a robust default, but they do not establish that it is optimal for every model.

The preferred tapering profile may depend on properties such as model depth, hidden dimension, the fraction of parameters allocated to the MLP, the token-mixing architecture, or the training budget. In particular, the best schedule or endpoint ratio may shift as models scale, potentially yielding gains beyond those reported here. A broader study that sweeps tapering configurations across model sizes and architectural families would help characterize these interactions and determine whether a single transferable schedule is sufficient or architecture- and scale-specific configurations are preferable.

7 Discussion and Conclusion

We introduced *Tapered Language Models* (TLMs), in which MLP width decreases monotonically across depth under a fixed total budget. A controlled comparison shows that the direction of allocation matters: front-loading MLP capacity improves perplexity, while back-loading it, the direction taken by prior layer-wise scaling, hurts. A smooth cosine taper, selected once on a 440M Transformer, improves average downstream accuracy and perplexity relative to uniform-width baselines for three architectures spanning softmax attention, gated attention, and neural long-term memory, at 760M and 1.3B parameters, with matched parameter count and FLOPs and modest throughput and memory overhead. A layer-wise analysis of pretrained transformers shows that MLP outputs become more aligned with the residual stream through the second half of the network, consistent with the tapering direction.

We see these results as a tested guideline for depth-wise capacity allocation within the scales and architectures studied. Whether similar gains extend to other per-layer dimensions, to larger scales, and to other model families, including vision transformers (Dosovitskiy et al., 2020), diffusion transformers (Peebles & Xie, 2023), and multimodal models (Radford et al., 2021; Liu et al., 2023), which also allocate capacity approximately uniformly across depth, are open empirical questions.

AI USE STATEMENT

We used generative AI tools only for readability editing of the manuscript. All edits were reviewed by the authors, who take full responsibility for the content of this paper.

REFERENCES

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, et al. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *Advances in Neural Information Processing Systems*, 38:96572–96617, 2026.

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

Reza Bayat, Ali Rahimi-Kalahroudi, Mohammad Pezeshki, Sarath Chandar, and Pascal Vincent. Steering large language model activations in sparse spaces. *arXiv preprint arXiv:2503.00177*, 2025.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2024.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025a.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. *arXiv preprint arXiv:2512.24695*, 2025b.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. *arXiv preprint arXiv:2504.13173*, 2025c.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Akhiad Bercovich, Tomer Ronen, Talor Abramovich, Nir Ailon, Nave Assaf, Mohammad Dabbah, Ido Galil, Amnon Geifman, Yonatan Geifman, Izhak Golan, et al. Puzzle: Distillation-based nas for inference-optimized llms. *arXiv preprint arXiv:2411.19146*, 2024.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pp. 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in neural information processing systems*, 33:4271–4282, 2020.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJg7KhVKPH.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1): 5232–5270, 2022.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 7657–7666, 2021.

Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. Liquid structural state-space models. *arXiv preprint arXiv:2209.12951*, 2022.

Jiaxi Hu, Yongqi Pan, Jusen Du, Disen Lan, Tang Tang, Qingsong Wen, Yuxuan Liang, and Weigao Sun. Improving bilinear rnn with closed-loop control. *Advances in Neural Information Processing Systems*, 38:107480–107506, 2026.

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.

Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. Matformer: Nested transformer for elastic inference. *Advances in Neural Information Processing Systems*, 37:140535–140564, 2024.

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference? *arXiv preprint arXiv:2406.19384*, 2024.

Zeman Li, Ali Behrouz, Yuan Deng, Peilin Zhong, Praneeth Kacham, Mahdi Karami, Meisam Razaviyayn, and Vahab Mirrokni. TNT: Improving chunkwise training for test-time memorization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rajioNWfRs.

Gui Ling, Ziyang Wang, Yuliang Yan, and Qingwen Liu. Slimgpt: Layer-wise structured pruning for large language models. *Advances in Neural Information Processing Systems*, 37:107112–107137, 2024.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. *arXiv preprint arXiv:2407.14207*, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

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.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

William Merrill, Yanhong Li, Tyler Romero, Anej Svete, Caia Costello, Pradeep Dasigi, Dirk Groeneveld, David Heineman, Bailey Kuehl, Nathan Lambert, et al. Olmo hybrid: From theory to practice and back. *arXiv preprint arXiv:2604.03444*, 2026.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International conference on machine learning*, pp. 26670–26698. PMLR, 2023.

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

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The FineWeb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *Findings of the association for computational linguistics: EMNLP 2023*, pp. 14048–14077, 2023.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, et al. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Haowen Hou, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, et al. Rwkv-7" goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*, 2025.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Liliang Ren, Yang Liu, Yadong Lu, Chen Liang, Weizhu Chen, et al. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *International Conference on Learning Representations*, volume 2025, pp. 53551–53575, 2025.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pp. 9355–9366. PMLR, 2021.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *arXiv preprint arXiv: 2207.07061*, 2022.

Pratyusha Sharma, Jordan T Ash, and Dipendra Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. *arXiv preprint arXiv:2312.13558*, 2023.

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

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Improving state-tracking in linear RNNs via householder products. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=SoRiaijTGr.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*, 2025.

Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Matteo Tiezzi, Michele Casoni, Alessandro Betti, Tommaso Guidi, Marco Gori, and Stefano Melacci. On the resurgence of recurrent models for long sequences: Survey and research opportunities in the transformer era. *arXiv preprint arXiv:2402.08132*, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 5797–5808, 2019.

Ke Alexander Wang, Jiaxin Shi, and Emily B Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

A Long-Context Retrieval on Needle-in-a-Haystack

We evaluate long-context retrieval on Needle-in-a-Haystack (NIAH) to verify that redistributing MLP capacity across depth does not degrade long-context behavior. We use three single-needle variants of increasing difficulty (S-NIAH-1 retrieves a passkey, S-NIAH-2 a numerical value, and S-NIAH-3 a UUID) and a multi-query variant (MQ-NIAH), each evaluated at three context lengths (4K, 8K, and 16K). Results are in Table 4. Tapered models match or slightly exceed their uniform counterparts in most cells, with no systematic degradation; most differences are within a fraction of a point.

Table 4: **Long-context retrieval results.** Needle-in-a-Haystack experiments with three single-needle variants of increasing difficulty (S-NIAH-1 retrieves a passkey, S-NIAH-2 a numerical value, and S-NIAH-3 a UUID) and a multi-query variant (MQ-NIAH), evaluated at three context lengths. All entries are retrieval accuracy; higher is better. Tapered rows are shaded.

	S-NIAH-1 (passkey)			S-NIAH-2 (number)			S-NIAH-3 (UUID)			MQ-NIAH (multi-query)		
Model	4K	8K	16K	4K	8K	16K	4K	8K	16K	4K	8K	16K
Transformer++	96.4	84.6	82.4	100	99.4	95.8	80.8	72.4	47.2	56.8	45.4	27.2
+ Tapered	96.4	84.8	82.8	100	99.2	96.4	81.4	72.4	47.8	57.4	45.4	27.6
Gated Attention	97.2	86.4	84.0	100	99.8	95.8	81.2	72.2	47.4	57.2	46.0	27.2
+ Tapered	97.2	86.8	84.0	100	99.6	96.2	81.2	72.4	47.8	57.8	46.2	27.2
Titans	100	100	100	99.2	83.8	67.8	74.4	38.8	18.2	21.2	19.2	11.8
+ Tapered	100	100	100	99.2	84.2	68.4	74.2	39.2	20.4	21.8	21.0	12.4