INVESTIGATING FFN PARAMETER ALLOCATION VIA CROSS-LAYER SHARING IN TRANSFORMER LMS

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper studies feed-forward network (FFN) parameter sharing in Transformer-based language models. We share FFN parameters across a contiguous middle block and reallocate the saved budget to widen FFNs, retaining FFN computation in every layer. A broad search using models with 570M non-embedding parameters identifies many configurations with lower average downstream task loss than a standard Transformer baseline. Improvements concentrate in late-heavy configurations that retain more layer-specific FFNs in the late layers. Mirrored comparisons with identical parameter counts and FLOPs within each pair also favor late-heavy allocation on average task loss in most cases. The selected configuration improves average downstream accuracy at the 2B scale without retuning the sharing configuration. These results highlight the importance of where independently parameterized FFNs are retained and suggest that sharing and reallocation can improve downstream performance at a nearly matched parameter budget.

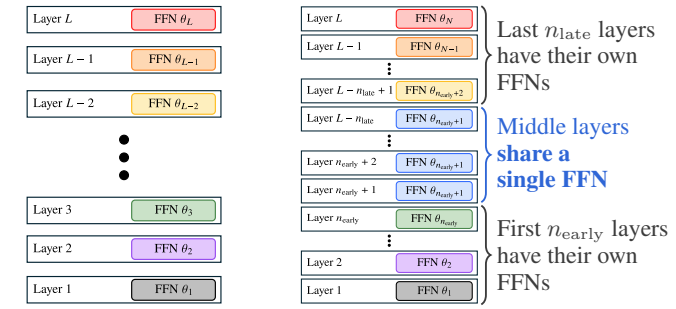
1 INTRODUCTION

Transformer-based language models (hereafter, "Transformer LMs") (Vaswani et al., 2017) typically consist of stacked Transformer layers, each containing a self-attention module and a feed-forward network (FFN). In standard Transformer architectures, each layer has its own FFN of identical size (Figure 1(a)). Although this uniform design is simple and effective, it remains unclear whether assigning an independent FFN of equal capacity to every layer is optimal, given that FFNs account for a large fraction of model parameters.

Prior work suggests that both FFNs and Transformer layers play depth-dependent roles. FFNs have been interpreted as key-value memories and shown to mediate factual associations (Geva et al., 2021; Meng et al., 2022). Lower layers tend to capture lexical or syntactic information, whereas higher layers encode more semantic information (Jawahar et al., 2019; Tenney et al., 2019; Geva et al., 2021; 2022). Recent analyses have characterized the middle layers as a region in which representations change little from one layer to the next (Jiang et al., 2025; Shibata et al., 2026) and individual layers are functionally redundant (Lad et al., 2025; Sun et al., 2025; Men et al., 2025), unlike early and late layers, which play more important roles in the behavior of the model. Similarly, a previous study has shown that reallocating FFN capacity to a consecutive block of middle layers can outperform standard uniform configurations under a comparable parameter budget (Ikeda et al., 2025). These findings point to layer position as an important factor in FFN parameter allocation.

This study examines how the placement of shared and layer-specific FFNs affects performance under a nearly matched parameter budget. Unlike the previous study, which removed FFNs from selected layers and reallocated the saved parameters to the remaining FFNs (Ikeda et al., 2025), we retain FFN computation in every layer. We study a Sandwich-style FFN sharing scheme: zero or more early layers and zero or more late layers keep layer-specific FFN parameters, while the contiguous block between them shares a single FFN parameter set (Figure 1(b)). This structure is motivated by the observation that the middle layers alter representations only slightly and are largely interchangeable, while early and late layers play more distinctive roles (§ 3.2). We further reallocate the saved parameter budget to widen all distinct FFN parameter sets, including the shared one (Figure 1(c)). This setup allows us to test whether sharing and redistributing FFN parameters can improve performance while keeping the total parameter budget nearly the same.

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.



(a) Standard Transformer (b) ***Sandwich-style FFN-shared model*** (c) Parameter budget

	Larger in
# distinct FFNs	(a) Standard
FFN width	(b) *Sandwich*
Total params	≈ **Matched** (✓)

Figure 1: Comparison of (a) a standard Transformer and (b) the Sandwich-style FFN-shared model. The standard Transformer assigns a layer-specific FFN (distinct colors) to every layer, whereas the Sandwich-style FFN-shared model shares a single FFN (blue) across the middle layers and retains layer-specific FFNs only in the first n_{early} and last n_{late} layers. (c) Sharing reduces the number of distinct FFN parameter sets, and the saved parameters are used to widen every FFN, so that the total parameter count approximately matches that of the standard Transformer.

We first conduct a broad search over Sandwich-style sharing configurations using Transformer LMs pretrained from scratch with approximately 570M non-embedding parameters, evaluating both held-out pretraining loss and downstream task loss. Many configurations reduce average downstream task loss relative to the baseline, with gains concentrated in late-heavy settings that retain more layer-specific FFNs near the output. We also compare mirrored configurations that swap the early- and late-layer counts while keeping parameter counts and FLOPs identical within each pair. These comparisons favor late-heavy allocation on average task loss in most cases, highlighting the importance of where independently parameterized FFNs are retained. Finally, we apply the best late-heavy configuration from the search to a larger model scale and a different resource constraint without retuning it, and test whether it retains its benefit over the baseline. It achieves higher average downstream accuracy than the baseline at the 2B scale, and under matched FLOPs, it roughly halves the number of non-embedding parameters while largely preserving average task loss.

We address four questions: (1) how FFN sharing and reallocation affect held-out pretraining loss and downstream task loss; (2) whether the parameter budget saved through sharing can be effectively reused to widen FFNs; (3) how the placement of layer-specific FFNs affects performance when parameter counts and FLOPs are matched within mirrored pairs; and (4) whether a configuration selected at the 570M scale retains its benefit over the baseline across model scales and resource constraints without retuning the sharing configuration.

2 Related Work

Functional Analysis of Feed-Forward Networks Several studies suggest that FFNs in pretrained Transformer LMs store linguistic and factual knowledge and mediate factual associations (Geva et al., 2021; Dai et al., 2022; Meng et al., 2022). Geva et al. (2021) interpreted FFNs as key-value memories, and subsequent work showed that stored knowledge and FFN contributions vary across layers (Geva et al., 2022; 2023; Meng et al., 2022; Lad et al., 2025). These findings motivate our question of whether FFN parameters should be allocated non-uniformly across layers.

Cross-Layer Parameter Sharing in Transformer Architectures Cross-layer parameter sharing has been widely studied in Transformer architectures. Prior work has shared a single parameter set across all layers (Lan et al., 2020), assigned a limited number of parameter sets to layers by a fixed rule (Takase & Kiyono, 2023; Ge et al., 2022; Bae et al., 2025), restricted sharing to specific components or layer ranges (Reid et al., 2021; Pires et al., 2023), or learned sharing patterns through routers (Tan et al., 2023; Csordás et al., 2024; Gong et al., 2024). Another line of work repeatedly applies the same parameters to increase the effective depth and computation beyond the number of layers (Dehghani et al., 2019; Geiping et al., 2025). In these studies, the sharing pattern is either given by a global rule or learned end-to-end. In contrast, we hold the number of layers and the parameter

budget fixed and treat the depth at which FFNs are shared as the variable under study, asking which layers can share a single FFN and which require layer-specific FFN parameters.

Parameter Allocation across Depth We share with several recent studies the premise that the standard design, which allocates parameters uniformly across depth, is not necessarily optimal (Ikeda et al., 2025; Wu et al., 2026; Bayat et al., 2026). These studies realize non-uniform capacity allocation by varying the width of each layer with depth, and report improvements over uniform baselines under a comparable parameter budget. In contrast, we keep the FFN width uniform across all layers and vary only which layers retain layer-specific FFN parameters. Whereas these studies ask how much width each layer requires, we ask how the dependence on layer-specific FFN parameterization varies with depth, and the two questions carry different design implications.

3 Model Description

3.1 Standard Transformer Architecture

In the following, we assume a Llama-style (Touvron et al., 2023) Pre-Layer Normalization (Pre-LN) Transformer (Xiong et al., 2020) equipped with SwiGLU (Shazeer, 2020) and RMSNorm (Zhang & Sennrich, 2019). The Transformer architecture (Vaswani et al., 2017) is built by stacking Transformer layers, each of which contains two subcomponents: self-attention and an FFN. Within each layer, two branches split off from the residual stream in turn, one for self-attention and the other for the FFN. Each branch processes the hidden state with its component and adds the result back to the residual stream via a residual connection. Hereafter, we refer to the stream that passes through the pre-FFN RMSNorm and the FFN as the *FFN stream*.

For a standard L-layer Transformer, let θ_l denote the set of learnable parameters of the FFN stream at the l-th layer ($1 \le l \le L$). Let d be the hidden state dimension and x the input hidden state vector. The FFN stream at the l-th layer, $\mathrm{FFN}(x;\theta_l)$, is formulated as:

$$\mathrm{FFN}(x;\theta_l) = W^l_{\mathrm{down}}\big(\mathrm{Swish}(W^l_{\mathrm{gate}}x') \odot W^l_{\mathrm{up}}x'\big), \quad \text{where } x' = \mathrm{RMSNorm}(x;\gamma^l).$$

Here, let $d^{\mathrm{base}}_{\mathrm{FF}}$ ($d^{\mathrm{base}}_{\mathrm{FF}} > d$) denote the FFN intermediate dimension. Then, $W^l_{\mathrm{gate}}, W^l_{\mathrm{up}} \in \mathbb{R}^{d^{\mathrm{base}}_{\mathrm{FF}} \times d}$ and $W^l_{\mathrm{down}} \in \mathbb{R}^{d \times d^{\mathrm{base}}_{\mathrm{FF}}}$ are the FFN weight matrices, and $\mathrm{Swish}(\cdot)$ is the activation function used in SwiGLU (Shazeer, 2020). In addition, $\mathrm{RMSNorm}(\cdot;\gamma^l)$ denotes the pre-FFN layer normalization with a learnable scaling vector $\gamma^l \in \mathbb{R}^d$ (Zhang & Sennrich, 2019). Accordingly, the set of learnable parameters in the l-th FFN stream is $\theta_l = \{W^l_{\mathrm{down}}, W^l_{\mathrm{gate}}, W^l_{\mathrm{up}}, \gamma^l\}$, containing $3dd^{\mathrm{base}}_{\mathrm{FF}} + d$ parameters. Because every layer of an L-layer Transformer holds independently learnable parameters (Figure 1(a)), the pool of independent parameter sets for the FFN streams across all L layers, $\Theta^{\mathrm{FF}}_{\mathrm{base}} = \{\theta_1, \ldots, \theta_L\}$, contains L distinct elements.

3.2 Sandwich-style FFN-shared Model

The parameter-sharing scheme we investigate redistributes FFN parameters non-uniformly across layers under an approximately matched total parameter budget, by sharing the FFN stream across a subset of layers. In standard cross-layer parameter sharing, multiple layers reference a single parameter set, which reduces the number of independent parameter sets and hence the total parameter count. To compensate for this reduction, we redirect the saved budget to expanding the FFN intermediate dimension, as detailed below. In this sense, our approach can be viewed as a layerwise reallocation of the parameter budget, which enhances the representational capacity of layers with layer-specific FFN parameters.

Although FFN parameters can be shared across layers in various ways, we focus on a specific design in which zero or more early layers and zero or more late layers retain their own FFN parameters, while the remaining contiguous middle layers share a single set of FFN parameters. We refer to this design as *Sandwich-style FFN sharing*[1] (see Figure 1(b)). This design choice is motivated by prior work on pretrained language models reporting that representations change little between adjacent middle

[1]We use the term Sandwich-style even for degenerate cases in which one or both sides contain no layer-specific layers, because it intuitively describes the concept.

layers (Jiang et al., 2025; Shibata et al., 2026) and that the middle layers are robust to the removal or reordering of individual layers (Lad et al., 2025; Sun et al., 2025; Men et al., 2025), whereas early and late layers play distinctly different roles. Together, these observations lead us to hypothesize that sharing an FFN across the middle layers costs little. We further posit that the parameter budget saved by this sharing is more valuable in the early and late layers, so that reallocating it to their layer-specific FFNs can improve performance.

In an L-layer Sandwich-style FFN-shared model, let N ($N < L$) denote the number of independent parameter sets over the FFN streams of the entire model, where a smaller N induces sharing across more layers. The pool of independent parameter sets in the Sandwich-style FFN-shared model is then $\Theta^{\mathrm{FF}}_{\mathrm{share}} = \{\theta_1, \ldots, \theta_N\}$. Let n_{early} and n_{late} denote the number of layers, counted from the first layer and from the final layer, respectively, that retain their own FFN stream parameters, while the remaining middle layers share a single parameter set. Therefore, $N = n_{\mathrm{early}} + n_{\mathrm{late}} + 1$ holds. Formally, the FFN stream at layer l is written as $\mathrm{FFN}(x; \theta_{\phi(l)})$, where the mapping $\phi : \{1, \ldots, L\} \to \{1, \ldots, N\}$ from the layer index l to the parameter index is defined as:

$$\phi(l) = \begin{cases} l & \text{if } l \le n_{\mathrm{early}} \\ n_{\mathrm{early}} + 1 & \text{if } n_{\mathrm{early}} < l \le L - n_{\mathrm{late}} \\ l - (L - N) & \text{if } l > L - n_{\mathrm{late}} \end{cases}$$

Because $\theta_{\phi(l)}$ includes the scaling vector $\gamma^{\phi(l)}$, the layers in the shared block share the pre-FFN RMSNorm as well as the FFN weight matrices. Any variant of an L-layer Sandwich-style FFN-shared model is characterized by a pair of non-negative integers $(n_{\mathrm{early}}, n_{\mathrm{late}})$ satisfying $n_{\mathrm{early}} + n_{\mathrm{late}} \le L - 2$, which ensures that the shared middle block spans at least two layers, so that genuine cross-layer parameter sharing takes place.

Parameter Budget Adjustment In the Sandwich-style FFN-shared model, we offset the parameter reduction due to sharing by expanding the FFN intermediate dimension from $d^{\mathrm{base}}_{\mathrm{FF}}$ to $d^{\mathrm{share}}_{\mathrm{FF}}$. That is, we choose $d^{\mathrm{share}}_{\mathrm{FF}}$ so that the number of parameters in $\Theta^{\mathrm{FF}}_{\mathrm{share}}$ approximately matches that in $\Theta^{\mathrm{FF}}_{\mathrm{base}}$. Because one FFN stream contains $3dd_{\mathrm{FF}} + d$ parameters as mentioned in § 3.1, this condition reduces to approximately solving $(3dd^{\mathrm{base}}_{\mathrm{FF}} + d)L = (3dd^{\mathrm{share}}_{\mathrm{FF}} + d)N$. Accordingly, $d^{\mathrm{share}}_{\mathrm{FF}}$ is defined as follows[2]:

$$d^{\mathrm{share}}_{\mathrm{FF}} = \left\lfloor \frac{L}{N} d^{\mathrm{base}}_{\mathrm{FF}} + \frac{L - N}{3N} \right\rfloor$$

Because FFN computation is still performed at every layer, expanding the intermediate dimension to $d^{\mathrm{share}}_{\mathrm{FF}}$ increases the per-token FLOPs in both training and inference[3]. The search in § 4 investigates which FFN parameter allocations achieve high parameter efficiency while keeping the number of parameters approximately fixed. These comparisons thus control for the memory required to store the weights (together with their gradients and optimizer states during training) but not for the FLOPs. We compare against the baseline under matched FLOPs in § 5.2 and discuss the relationship between additional computation and performance gains in § 6.

4 Exploring Better FFN Parameter Allocation

In this section, we conduct a broad search over Sandwich-style FFN sharing configurations to identify allocation strategies that outperform a standard Transformer at a comparable parameter budget.

4.1 Experimental Settings

Baseline Model Configurations We adopt a Llama-style Pre-LN Transformer architecture (Touvron et al., 2023) as our base architecture. We set the number of layers to $L = 24$ and the hidden state dimension to $d = 1280$. Hereafter, we refer to this standard Transformer model, in which every layer holds independently trainable parameters, as the *baseline* model. The FFN intermediate dimension of the baseline model is set to $d^{\mathrm{base}}_{\mathrm{FF}} = 4480$. Consequently, the total number of parameters across all Transformer layers, excluding the embedding, is 570 million. For the tokenizer, we use the Llama 3.1 tokenizer, which has a vocabulary size of 128,256.

[2] In practice, we use $32\lceil d^{\mathrm{share}}_{\mathrm{FF}}/32 \rceil$ for efficient GPU matrix operations.

[3] Appendix E lists the theoretical FLOPs for each N.

Sandwich-style FFN-shared Model Configurations For $L = 24$, a total of 276 variants are definable, forming the triangular region of the $(n_{\text{early}}, n_{\text{late}})$ grid satisfying $n_{\text{early}} + n_{\text{late}} \leq 22$ (Figure 2). Since pretraining all 276 variants from scratch is computationally expensive, we instead sample configurations from this grid. Specifically, both n_{early} and n_{late} take values in $\{0, \ldots, 6\} \cup \{8, 10, \ldots, 22\}$. That is, we sample densely in the extreme high-sharing region ($n_{\text{early}}, n_{\text{late}} \leq 6$) and more sparsely elsewhere. This sampling spans the full extent of the definable search space, allowing us to examine where within it well-performing regions lie. The resulting 147 Sandwich-style FFN-shared variants each have a parameter count closely matching that of the baseline model[4].

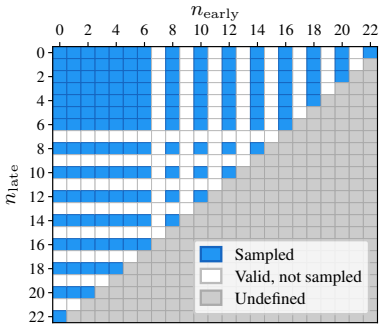


Figure 2: Definable $(n_{\text{early}}, n_{\text{late}})$ search space for $L = 24$. Blue cells indicate the 147 pretrained configurations.

Pretraining We pretrain all 147 Sandwich-style FFN-shared variants and the baseline from scratch with causal language modeling on FineWeb-Edu (Penedo et al., 2024). Following the Chinchilla-optimal recipe (Hoffmann et al., 2022), each model is trained on approximately 18B tokens, roughly $20\times$ its parameter count, including the embedding. We use AdamW (Loshchilov & Hutter, 2019) with the warmup-stable-decay (WSD) learning rate schedule (Hu et al., 2024; Wen et al., 2025) (peak learning rate $9\text{e}{-4}$). For a fair comparison, all Sandwich-style FFN-shared variants and the baseline are pretrained with identical hyperparameters, and the full settings are listed in Appendix B.

Loss Evaluation We evaluate the pretrained models with two metrics: *validation loss* and *task loss*. Validation loss is the per-token negative log-likelihood (NLL) on held-out samples from the pretraining corpus. Task loss (Nakamura et al., 2026; Brandfonbrener et al., 2025; Grattafiori et al., 2024) is the per-token NLL of the correct answer given its 5-shot prompt, averaged over the samples of a downstream task. We use task loss as our primary metric because it is continuous and allows for finer-grained comparisons. We evaluate task loss on seven tasks: HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), Natural Questions (Kwiatkowski et al., 2019), and TriviaQA (Joshi et al., 2017). All evaluations use the final checkpoint, and Appendix C details the procedure. We report the per-token NLL difference from the baseline, $\Delta\text{NLL}_{\text{baseline}} = \text{NLL}_{\text{baseline}} - \text{NLL}_{\text{target}}$ (in nats), where positive values indicate improvement.

4.2 Results

Figure 3 reports $\Delta\text{NLL}_{\text{baseline}}$ for all 147 Sandwich-style FFN-shared variants pretrained at the 570M-parameter scale.[5] Enlarged heatmaps for all nine metrics, including the five downstream tasks not shown in the figure, are given in Appendix I.

Trends in Downstream Task Loss and Validation Loss Sandwich-style FFN sharing improves average downstream task loss across a broad range of configurations. As shown in Table 1[6], on average task loss, 126 out of 147 configurations (85.7%) improve over the baseline, with an average improvement of 0.088 among the 126 improved configurations. A similar trend holds at the level of individual tasks: in 6 out of 7 tasks, the majority of configurations improve over the baseline. On ARC-Challenge in particular, 131/147 (89.1%) configurations improve, with the largest improvement reaching 0.438. By contrast, on validation loss, only 27 out of 147 configurations improve. As Figure 3(a) makes clear, this gap arises because the region of improvement is narrower for validation loss than for task loss. This discrepancy is consistent with prior findings that pretraining loss alone does not fully determine downstream performance (Nakamura et al., 2026; Liu et al., 2023). These improvements do not appear in isolation but form a contiguous region on the sampled grid. The blue

[4] See Appendix A for all $(n_{\text{early}}, n_{\text{late}})$ pairs examined with their precise configurations.

[5] See Appendix F for the absolute NLL values of the baseline model.

[6] In all tables, **Val.** denotes validation loss and **Task Avg.** denotes average task loss. Task names are abbreviated as follows: **HS** (HellaSwag), **ARC-E** (ARC-Easy), **ARC-C** (ARC-Challenge), **OBQA** (OpenBookQA), **NQ** (Natural Questions), and **TQA** (TriviaQA).

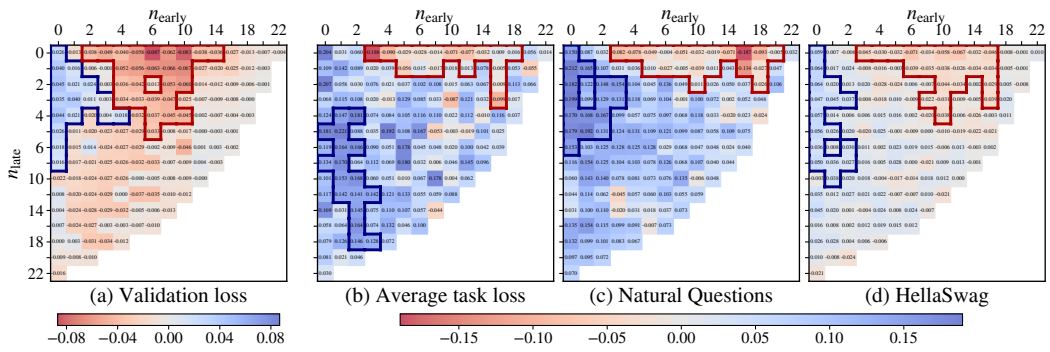


Figure 3: Heatmaps of $\Delta\text{NLL}_{\text{baseline}}$ for all 147 Sandwich-style FFN-shared models (570M scale). Panel (a) shows validation loss, (b) task loss averaged over seven downstream tasks, and (c)–(d) two of the individual tasks. The horizontal axis is n_{early} and the vertical axis is n_{late}, and the blank region (lower right) corresponds to undefined configurations. Blue, red, and white indicate improvement, degradation, and parity over the baseline, respectively, with an independent color scale for (a) and a shared one for (b)–(d). The blue and red outlines in each heatmap mark the largest 4-connected component among the top-30 and bottom-30 cells, respectively, where connectivity is defined on the sampled grid, with consecutive sampled axis values treated as adjacent.

Table 1: Summary of $\Delta\text{NLL}_{\text{baseline}}$ across the 147 Sandwich-style FFN-shared variants. *# impr.* counts configurations with positive $\Delta\text{NLL}_{\text{baseline}}$, and *Max* Δ and *Mean* Δ *(impr.)* give the maximum and the mean over these improved configurations.

	Val.	Task Avg.	NQ	HS	ARC-E	ARC-C	OBQA	PIQA	TQA
# impr. / 147	27	126	123	79	125	131	111	126	69
Max Δ	0.045	0.221	0.212	0.064	0.471	0.438	0.191	0.186	0.181
Mean Δ (impr.)	0.017	0.088	0.090	0.020	0.192	0.194	0.081	0.071	0.070

outlines in each panel of Figure 3 indicate the largest 4-connected component formed by the top-30 cells, confirming that the improvements emerge as a broad contiguous region. This contiguity is consistent with a systematic allocation pattern rather than an isolated favorable configuration.

The Improvement Concentrates in the Late-Heavy Region We partition the search space along the diagonal $n_{\text{early}} = n_{\text{late}}$ into two regions, the *late-heavy* region satisfying $n_{\text{late}} \geq n_{\text{early}}$ (78 configurations) and the *early-heavy* region satisfying $n_{\text{late}} < n_{\text{early}}$ (69 configurations). For simplicity, we include the 9 balanced configurations on the diagonal in the late-heavy region.[7] Improvements are not uniformly distributed across the search space but concentrate in the late-heavy region. This is directly visible in Figure 3: in all four panels, as well as in the heatmaps of the remaining five tasks (Appendix I), the cluster of top-30 cells (blue outlines) lies almost entirely within the late-heavy region, whereas the cluster of bottom-30 cells (red outlines) lies within the early-heavy region. In other words, the best- and worst-performing configurations cluster on opposite sides of the diagonal. Table 2 confirms this trend with region-level statistics. The mean $\Delta\text{NLL}_{\text{baseline}}$ on average task loss is 0.101 in the late-heavy region but only 0.031 in the early-heavy region, a more than three-fold difference. Likewise, 98.7% of the late-heavy configurations outperform the baseline, compared with 71.0% of the early-heavy ones. Validation loss shows the same tendency: configurations that outperform the baseline are found mainly in the late-heavy region (30.8% vs. 4.3%). Moreover, for all 7 downstream tasks, the late-heavy region shows both a larger mean improvement and a higher proportion of improved configurations.

Mirrored-Pair Comparison Late-heavy configurations clearly outperform early-heavy ones even when compared with their mirrored counterparts, which have exactly the same parameter count and

[7]Excluding them does not change the conclusions below. The remaining 69 configurations with $n_{\text{late}} > n_{\text{early}}$ have a mean $\Delta\text{NLL}_{\text{baseline}}$ of 0.105 on average task loss, and 98.6% of them outperform the baseline.

Table 2: Comparison of early-heavy and late-heavy regions by metric. *Mean* Δ is the mean $\Delta\mathrm{NLL}_{\mathrm{baseline}}$ in each region, and *% improved* is the fraction of configurations with positive $\Delta\mathrm{NLL}_{\mathrm{baseline}}$. *Mirror-pair win* is the fraction of the 69 mirrored pairs $\{(a,b),(b,a)\}$ in which the late-heavy configuration has the larger $\Delta\mathrm{NLL}_{\mathrm{baseline}}$.

		Val.	Task Avg.	NQ	HS	ARC-E	ARC-C	OBQA	PIQA	TQA
Mean Δ	early-heavy	-0.025	0.031	0.032	-0.014	0.081	0.107	0.022	0.030	-0.040
	late-heavy	$\mathbf{-0.007}$	**0.101**	**0.101**	**0.015**	**0.204**	**0.214**	**0.073**	**0.078**	**0.022**
% improved	early-heavy	4.3	71.0	71.0	26.1	72.5	78.3	69.6	79.7	31.9
	late-heavy	**30.8**	**98.7**	**94.9**	**78.2**	**96.2**	**98.7**	**80.8**	**91.0**	**60.3**
Mirror-pair win (%)		62.3	84.1	82.6	78.3	75.4	76.8	73.9	76.8	69.6

Table 3: Accuracy (%) of the Sandwich-style FFN-shared model with $(n_{\mathrm{early}}, n_{\mathrm{late}}) = (1,5)$ (*Sand.*) and the baseline (*Base.*) at the 2B-parameter scale. Δ_{acc} is Sand. $-$ Base.; *lower* and *upper* give its 95 % paired bootstrap percentile interval (10,000 replicates), and * marks intervals excluding zero.

	Selection set								*Held-out set*					
	ARC-C	ARC-E	HS	NQ	OBQA	PIQA	TQA	Avg.	LAMB	RACE	SciQ	SIQA	WG	Avg.
Sand.	39.6	73.1	45.3	7.5	27.0	73.9	2.1	38.4	38.8	35.8	93.7	44.7	59.9	54.6
Base.	38.6	73.5	44.0	6.0	27.4	74.3	2.0	38.0	38.0	35.3	93.5	44.0	57.5	53.7
Δ_{acc}	$+1.0$	-0.4	$\mathbf{+1.4}^*$	$\mathbf{+1.5}^*$	-0.4	-0.4	$+0.1$	$+0.4$	$+0.8$	$+0.5$	$+0.2$	$+0.7$	$+2.4$	$\mathbf{+0.9}^*$
lower	-1.4	-1.9	$+0.9$	$+0.8$	-3.6	-1.8	-0.1	-0.2	-0.3	-1.6	-1.1	-0.7	-0.5	$+0.1$
upper	$+3.4$	$+1.0$	$+1.8$	$+2.3$	$+2.8$	$+0.9$	$+0.4$	$+1.0$	$+1.9$	$+2.6$	$+1.5$	$+2.1$	$+5.2$	$+1.7$
All 12 tasks average: Sandwich 45.1, Baseline 44.5, $\Delta_{\mathrm{acc}} = \mathbf{+0.6}^*$ $[+0.1, +1.1]$														

FLOPs. Because our sampling is symmetric about the diagonal, each early-heavy configuration (a,b) with $a > b$ has a mirrored late-heavy counterpart (b,a), yielding 69 pairs. A mirrored pair shares the same $N = a + b + 1$, the same FFN width, and the same shared-block length $L - a - b$. They differ only in the position of the shared block, which is shifted by $|a - b|$ layers between the two models. For example, the shared blocks of $(1,5)$ and $(5,1)$ both include layers 6–19, and the remaining four shared layers lie on the input side (layers 2–5) in $(1,5)$ but on the output side (layers 20–23) in $(5,1)$. Despite this minimal difference, the late-heavy configuration outperforms its mirror in 58 of the 69 pairs (84.1%) on average task loss, and in 69.6–82.6% of the pairs on each of the 7 downstream tasks (Table 2). This indicates that the advantage of late-heavy configurations stems not from differences in parameter count or FLOPs but from where the shared layers are placed.

5 Transferability across Scale and Resource Constraints

In § 4, we conducted the search with 570M-scale models under the resource constraint of an approximately fixed model-state memory budget and observed that performance varied depending on the placement of shared layers within mirrored pairs, despite identical parameter counts and FLOPs. However, repeating such a search for every setting is computationally expensive. In this section, we therefore reuse the best configuration from this search without retuning and examine whether it retains its benefit over the baseline when the model scale or the resource constraint changes. Specifically, we take $(n_{\mathrm{early}}, n_{\mathrm{late}}) = (1,5)$, which achieves the largest $\Delta\mathrm{NLL}_{\mathrm{baseline}}$ on average task loss (0.221) among the 147 configurations in the search, and evaluate it under two changes to the setting.[8] First, we change the model scale from § 4 while keeping its resource constraint, scaling the model up to 2B parameters with its model-state memory budget again nearly matched to that of the baseline (§ 5.1). Second, we change the resource constraint from § 4 while keeping its depth and width, matching the baseline exactly in FLOPs instead of model-state memory budget by applying only the sharing pattern without FFN expansion (§ 5.2).

[8]We further confirm that $(1,5)$ retains its benefit when the pretraining corpus is changed (Appendix J).

5.1 Scaling Up to 2B Parameters

The baseline has $L = 40$ and $d = 1920$, with 2.1B non-embedding parameters. The Sandwich-style FFN-shared model applies $(1, 5)$, with the FFN intermediate dimension set as in § 3.2 to nearly match this parameter count. Because $(1, 5)$ is applied as absolute layer counts rather than ratios, the shared middle block grows from 18 of 24 layers to 34 of 40. Appendices B and D detail the pretraining and evaluation settings. The main differences from the evaluation protocol of § 4 are that we evaluate accuracy, which becomes more stable at this scale, instead of task loss, and that we add 5 tasks not used for configuration selection (the *held-out set*) to the 7 tasks used for it (the *selection set*).

At the 2B scale, $(1, 5)$ also outperforms the baseline even with a substantially longer shared block (Table 3[9]). Its average accuracy over the 12 tasks is 0.6 pp higher than that of the baseline, and the 95% bootstrap percentile interval excludes zero. This improvement remains statistically significant on the held-out set (+0.9 pp), suggesting that it is not specific to the tasks used for configuration selection. At the task level, the trend is consistent across most tasks: $(1, 5)$ outperforms the baseline on 9 of the 12 tasks and shows no statistically significant degradation on the remaining 3.

5.2 Matching the Baseline FLOPs

Following the 570M-scale training and evaluation settings of § 4, we apply only the $(1, 5)$ sharing pattern and keep the FFN intermediate dimension at the baseline width $d_{\mathrm{FF}}^{\mathrm{base}}$. Because every layer then computes an FFN of the baseline width, the model exactly matches the baseline in training and inference FLOPs (Appendix E). FFN sharing, in contrast, reduces the non-embedding parameters from 570.2M to 277.7M (about 51%). This setting thus reverses the resource constraint of § 4, matching the FLOPs while letting the model-state memory change.

Table 4: $\Delta\mathrm{NLL}_{\mathrm{baseline}}$ of $(1, 5)$ without FFN expansion (570M).

Metric	$\Delta\mathrm{NLL}_{\mathrm{baseline}}$
Val. Loss	−0.097
Task Loss (Avg.)	**−0.029**
NQ	−0.052
HS	−0.063
ARC-E	**0.067**
ARC-C	**0.129**
OBQA	−0.158
PIQA	**−0.002**
TQA	−0.123

Under matched FLOPs, the benefit of $(1, 5)$ over the baseline shifts from performance to model-state memory. Although it does not outperform the baseline, it roughly halves the number of non-embedding parameters while increasing average task loss by only 0.029 (Table 4). At the very least, this degradation is smaller than that of 11 of the 147 configurations in § 4, all of which use the full parameter budget and more FLOPs than the baseline. Moreover, at the task level, it performs comparably to or better than the baseline on some tasks (ARC-Easy, ARC-Challenge, and PIQA).

Transferability and Operating Points Taken together, although its optimality in either setting is untested, the best configuration from the search retains a benefit over the baseline in both settings above, without retuning the sharing configuration. This benefit appears as different useful operating points depending on the resource constraint. Under a matched model-state memory budget (§ 4 and § 5.1), it improves performance at the cost of more computation. Under matched FLOPs, it instead substantially reduces model-state memory while largely preserving average task loss.

6 Discussion

Does Additional Computation Alone Explain the Gains? In the comparisons of § 4 and § 5.1, the parameter count, and hence the model-state memory, is matched to that of the baseline, whereas FFN expansion increases the FLOPs beyond those of the baseline. However, this additional computation alone does not explain the gains over the baseline. Figure 4 plots $\Delta\mathrm{NLL}_{\mathrm{baseline}}$ of the average task loss for the 147 configurations in § 4 against N. Configurations with the same N have identical FLOPs, which are shown relative to the baseline on the top axis. Although every configuration uses more FLOPs than the baseline, 21 of them underperform it, and 20 of these 21 are early-heavy. Moreover, 17 of the 22 values of N with multiple configurations include both improved and degraded configurations, and the Spearman rank correlation between the FLOPs ratio and the gain is only

[9] Abbreviations follow those in Table 1. Further abbreviations are **LAMB** (LAMBADA), **SIQA** (Social IQa), and **WG** (WinoGrande). **Avg.** here denotes average accuracy.

0.12. In contrast, within each N, late-heavy configurations generally lie above early-heavy ones (the best configuration is late-heavy for 20 of the 22 values of N), consistent with the mirrored-pair comparison in § 4.2. Therefore, the advantage over the baseline depends not only on the increase in FLOPs but also on the placement of layer-specific FFNs. However, although we do not observe an improvement in average task loss at matched FLOPs, the benefit in this setting is reduced model-state memory while maintaining a similar performance level (see § 5.2 for results under matched FLOPs).

Why Does Late-heavy Allocation Help? We view the performance change of a Sandwich-style FFN-shared model as the difference between the gain from widening FFNs and the cost of forcing a single FFN parameter set onto layers at different depths (Appendix G). Because both configurations in each mirrored pair have the same FFN width and are thus directly comparable, the late-heavy advantage in mirrored pairs (§ 4.2) suggests that sharing layers near the output is relatively less effective than sharing layers near the input. We offer two working hypotheses for this asymmetry. First, FFNs near the output may serve more depth-specific functions. Because late FFNs shape the final prediction by additively updating the vocabulary distribution (Geva et al., 2022; Lad et al., 2025), they may require different promotion and suppression of tokens at each depth, and tying them to a single parameter set can cause stronger interference than tying early FFNs. Second, mismatches introduced near the output may be harder to compensate for. Because subsequent layers partially correct the contributions of preceding layers (Patrawala et al., 2025), a mismatch from sharing near the input can be compensated for by the many later layers, whereas one near the output is more likely to persist in the final prediction. Both hypotheses remain untested, and verifying the mechanism is left to future work.

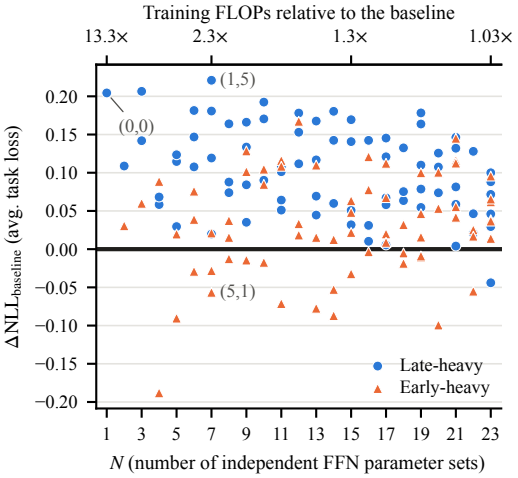


Figure 4: $\Delta\mathrm{NLL}_{\mathrm{baseline}}$ of the average task loss for all 147 Sandwich-style FFN-shared models (570M scale) against N. The thick line at zero marks the baseline.

Limitations This study had several limitations. First, our comprehensive experiments were limited to Llama-style causal Transformer LMs trained on FineWeb-Edu, and it remains unclear whether the same late-heavy tendency holds for other architectures, training corpora, or token budgets. Second, our search space was restricted to Sandwich-style FFN sharing, where a single contiguous block of layers shares FFN parameters. More general sharing patterns may yield further improvements. Third, the experiments in § 5 evaluated only the $(1, 5)$ configuration selected in § 4, so its optimality at the 2B scale or under matched FLOPs is untested. Finally, each configuration was pretrained with a single seed because training Transformer LMs from scratch is computationally expensive.

7 Conclusion

We studied Sandwich-style FFN sharing as a layerwise parameter allocation strategy for Transformer LMs. We shared FFN parameters across a contiguous middle block and reallocated the saved parameter budget to widen the remaining distinct FFN parameter sets. A broad 570M-scale search showed that many Sandwich-style configurations reduced average downstream task loss relative to the baseline, with gains concentrated in late-heavy settings that retain more layer-specific FFNs in the late layers. Mirrored comparisons at this scale, with identical parameter counts and FLOPs within each pair, favored late-heavy allocation on average task loss in most cases, highlighting the importance of where independently parameterized FFN streams are retained. Scaling the selected $(n_{\mathrm{early}}, n_{\mathrm{late}}) = (1, 5)$ configuration to the 2B regime further improved the average downstream accuracy relative to the baseline without retuning the sharing configuration. These results suggest that careful FFN sharing and reallocation can improve downstream performance at a nearly matched parameter budget. We hope that systematic architectural trials of this kind can help improve the fundamental performance of Transformer LMs.

AI USE STATEMENT

We used generative AI tools at several stages of the study and manuscript preparation.

We consulted an LLM about implementation approaches for modifying the publicly available Meta Lingua codebase (Videau et al., 2024) to implement Sandwich-style FFN sharing. As the authors are non-native English speakers, we used an LLM to help translate portions of drafts written in our native language into English. We also used generative AI tools to assist with creating and modifying figures, writing and editing code, searching for and summarizing the literature, identifying relevant work and locating bibliographic information for BibTeX entries, and editing the manuscript to improve readability.

Additionally, we used an LLM to check the internal consistency of the manuscript. This included feedback on how the experimental methodology was described and on whether the reported results supported the stated claims and interpretations. The authors assessed these suggestions and made all final scientific decisions.

We reviewed all AI-assisted work. We tested and reviewed the implementation to check that it behaved as intended, checked translated and edited text for accuracy, checked AI-assisted figures for consistency with the methods and reported results, and verified all references against their original sources. The authors take responsibility for the final content of this work, including all text, claims, code, and figures produced with the assistance of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, and we did not collect any new personal data. We pretrain all models on publicly available corpora (FineWeb-Edu and WebOrganizer Corpus-200B) and evaluate them on publicly available benchmarks, all of which we use in accordance with their licenses (Appendix K). We do not create any new datasets. We plan to release the pretrained models and code under licenses consistent with those of the training data and the existing code they build on. The released models are research models pretrained to validate the proposed method at the 570M and 2B scales. They have not undergone instruction tuning or safety alignment and are not intended for deployment. As with other language models trained on web text, they may reflect biases or harmful content present in the training data. Our contribution concerns how FFN parameters are allocated across layers, and it does not introduce new capabilities or methods that specifically enable harmful applications.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we provide the following information. The definition of the Sandwich-style FFN-shared model and the procedure for determining the FFN width that matches the parameter count of the baseline are given in § 3.2. The configurations of all models are listed in Appendix A, and the pretraining settings in Appendix B. The evaluation procedures are described in Appendix C (validation loss and task loss) and Appendix D (tasks, numbers of shots, checkpoint averaging, and significance testing), and the computation of theoretical FLOPs in Appendix E. The publicly available datasets and libraries used for training and evaluation, together with their sources, are listed in Appendix K. In addition, we submit anonymized code for training and evaluation as supplementary material, and we will release the pretrained models upon acceptance.

REFERENCES

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=WwpYSOkkCt.

Reza Bayat, Ali Behrouz, and Aaron Courville. Tapered language models, 2026. URL https://arxiv.org/abs/2606.23670.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

David Brandfonbrener, Nikhil Anand, Nikhil Vyas, Eran Malach, and Sham M. Kakade. Loss-to-loss prediction: Scaling laws for all datasets. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=1Avb4jYjLb. Featured Certification.

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, 2018. URL https://arxiv.org/abs/1803.05457.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D Manning. MoEUT: Mixture-of-experts universal transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ZxVrkm7Bjl.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.581. URL https://aclanthology.org/2022.acl-long.581/.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

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.

Tao Ge, Si-Qing Chen, and Furu Wei. EdgeFormer: A parameter-efficient transformer for on-device seq2seq generation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 10786–10798, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.741. URL https://aclanthology.org/2022.emnlp-main.741/.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=S3GhJooWIC.

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/.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.751. URL https://aclanthology.org/2023.emnlp-main.751/.

Zhuocheng Gong, Ang Lv, Jian Guan, Wei Wu, Huishuai Zhang, Minlie Huang, Dongyan Zhao, and Rui Yan. Mixture-of-modules: Reinventing transformers as dynamic assemblies of modules. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20924–20938, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1164. URL https://aclanthology.org/2024.emnlp-main.1164/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan,

Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi (Jack) Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu (Sid) Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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, Jack W. Rae, Oriol Vinyals, and L. Sifre.

Training compute-optimal large language models. *ArXiv*, abs/2203.15556, 2022. URL https://api.semanticscholar.org/CorpusID:247778764.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=3X2L2TFr0f.

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. In *Conference on Language Modeling, COLM 2025*, 2025. URL https://openreview.net/forum?id=oP3b5YBFoP.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1356. URL https://aclanthology.org/P19-1356/.

Jiachen Jiang, Jinxin Zhou, and Zhihui Zhu. Tracing representation progression: Analyzing and enhancing layer-wise similarity. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vVxeFSR4fU.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. Remarkable robustness of LLMs: Stages of inference? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Wxh5Xz7NpJ.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082. URL https://aclanthology.org/D17-1082/.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

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.

Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. Same pre-training loss, better downstream: Implicit bias matters for language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22188–22214. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/liu23ao.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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 Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1035. URL https://aclanthology.org/2025.findings-acl.1035/.

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.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Taishi Nakamura, Satoki Ishikawa, Masaki Kawamura, Takumi Okamoto, Daisuke Nohara, Jun Suzuki, and Rio Yokota. Optimal sparsity of mixture-of-experts language models for reasoning tasks. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=XFw2EPRUUR.

Yuto Nishida, Masaru Isonuma, and Yusuke Oda. Instability in downstream task performance during LLM pretraining. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 22883–22895, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1246. URL https://aclanthology.org/2025.findings-emnlp.1246/.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 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/.

Arjun Patrawala, Jiahai Feng, Erik Jones, and Jacob Steinhardt. LLM layers immediately correct each other. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=7DY7kB8wyZ.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

Telmo Pires, António Vilarinho Lopes, Yannick Assogba, and Hendra Setiawan. One wide feedforward is all you need. In Philipp Koehn, Barry Haddow, Tom Kocmi, and Christof Monz (eds.), *Proceedings of the Eighth Conference on Machine Translation*, pp. 1031–1044, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.wmt-1.98. URL https://aclanthology.org/2023.wmt-1.98/.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 4081–4090, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.344. URL https://aclanthology.org/2021.findings-emnlp.344/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *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, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

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

Keigo Shibata, Kazuki Yano, Ryosuke Takahashi, Jaesung Lee, Wataru Ikeda, and Jun Suzuki. Suppressing final layer hidden state jumps in transformer pretraining. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 1236–1262, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-386-9. doi: 10.18653/v1/2026.findings-eacl.64. URL https://aclanthology.org/2026.findings-eacl.64/.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/J.NEUCOM.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. Transformer layers as painters. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 25219–25227. AAAI Press, 2025. doi: 10.1609/AAAI.V39I24.34708. URL https://doi.org/10.1609/aaai.v39i24.34708.

Sho Takase and Shun Kiyono. Lessons on parameter sharing across layers in transformers. In Nafise Sadat Moosavi, Iryna Gurevych, Yufang Hou, Gyuwan Kim, Young Jin Kim, Tal Schuster, and Ameeta Agrawal (eds.), *Proceedings of the Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, pp. 78–90, Toronto, Canada (Hybrid), July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.sustainlp-1.5. URL https://aclanthology.org/2023.sustainlp-1.5/.

Shawn Tan, Yikang Shen, Zhenfang Chen, Aaron Courville, and Chuang Gan. Sparse universal transformer. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 169–179, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.12. URL https://aclanthology.org/2023.emnlp-main.12/.

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/.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023. doi: 10.48550/ARXIV.2302.13971. URL https://doi.org/10.48550/arXiv.2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Mathurin Videau, Badr Youbi Idrissi, Daniel Haziza, Luca Wehrstedt, Jade Copet, Olivier Teytaud, and David Lopez-Paz. Meta Lingua: A minimal PyTorch LLM training library, 2024. URL https://github.com/facebookresearch/lingua.

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

Kaiyue Wen, Zhiyuan Li, Jason S. Wang, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape view. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=m51BgoqvbP.

Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=boSqwdvJVC.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Zhaofeng Wu, Oliver Sieberling, Shawn Tan, Rameswar Panda, Yury Polyanskiy, and Yoon Kim. Variable-width transformers, 2026. URL https://arxiv.org/abs/2606.18246.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10524–10533. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/xiong20b.html.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 12360–12371, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Table 5: Pretraining settings of the 570M-scale and 2B-scale models. All Sandwich-style FFN-shared models and the baseline at the same scale share these settings.

	570M scale	2B scale
Training corpus	FineWeb-Edu	
Training tokens	17.8B	52.4B
Optimizer steps	34,000	25,000
Batch size (tokens)	524,288	2,097,152
Sequence length	1,024	1,024
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$)	
Weight decay	0.1	
Gradient clipping	1.0	
Learning rate schedule	WSD	
Peak learning rate	9e−4	3e−4
Warmup steps	1,000	
Decay phase	Final 10% of training	
Final learning rate	10% of the peak	
Precision	bf16 mixed precision	

A Detailed Model Configuration

570M-Scale Models Table 11 lists the complete set of model configurations used in the search experiment of § 4. For all 570M-scale models, we use standard multi-head attention with 10 attention heads (i.e., no grouped-query attention). Positional embeddings follow the RoPE scheme (Su et al., 2024) with a base frequency of 10,000. The vocabulary size of the Llama 3.1 tokenizer is 128,256.

2B-Scale Models Table 12 lists the model configurations used in the scaling experiment of § 5.1. For both the baseline and the Sandwich-style FFN-shared model, we use standard multi-head attention with 15 attention heads. All other architecture choices (RoPE, vocabulary) are identical to the 570M-scale setting.

Embedding and Parameter Counts At both scales, the input token embedding and the output projection (LM head) are not tied, and each contains $128{,}256 \times d$ parameters. The # Params (w/o emb.) column in Tables 11 and 12 excludes both of these matrices, and we refer to this count as the number of non-embedding parameters throughout the paper.

Computational Resources All experiments were conducted on `NVIDIA H100 80GB` GPUs. The 570M-scale search (147 variants plus the baseline) required approximately 21,300 GPU hours in total, and the three additional 570M-scale runs reported in the appendix (the baseline and $(1, 5)$ on WebOrganizer, and $(1, 5)$ without FFN expansion; Appendix J and G) required approximately 430 GPU hours. The 2B-scale experiments (one Sandwich-style variant and the baseline) required approximately 3,104 GPU hours in total.

B Pretraining Settings

Table 5 lists the pretraining settings of the 570M-scale search (§ 4) and the 2B-scale experiment (§ 5.1). All models were implemented and pretrained using the `Meta Lingua` framework (Videau et al., 2024), and all models at the same scale, including the baseline, share identical settings. The two scales share the optimizer and the learning rate schedule, and differ only in the peak learning rate, the batch size, and the number of optimizer steps. The WSD schedule warms up the learning rate linearly over the first 1,000 steps, keeps it constant, and then decays it to 10% of its peak value over the final 10% of training, with the reciprocal of the learning rate increasing linearly during the decay.

C Loss Evaluation Details

This section details the loss-based evaluation of the 570M-scale models in § 4. All evaluations use the final checkpoint (step $34{,}000$).

Validation Loss We compute validation loss on the first $1{,}000$ documents of a held-out split of FineWeb-Edu that is disjoint from the training data. Each document is truncated to its last 511 tokens, and we compute the NLL of every token except the first, which serves only as context, i.e., of up to 510 tokens per document. We average these NLLs within each document and then average the resulting per-token NLLs over the $1{,}000$ documents, so that each document contributes equally regardless of its length.

Task Loss We construct each task-loss sample from the 5-shot prompts of the `LM Evaluation Harness` (Gao et al., 2024): the prompt consists of five in-context examples followed by the test question, and the answer is the correct continuation. For the multiple-choice tasks (ARC-Easy, ARC-Challenge, HellaSwag, OpenBookQA, and PIQA), the answer is the correct choice; for Natural Questions and TriviaQA, which accept multiple reference answers, we use the first reference answer. We feed the concatenation of the prompt and the answer to the model and divide the NLL summed over the answer tokens by the number of answer tokens. The task loss of a task is the average of this per-token NLL over its samples, and the average task loss is the mean over the seven tasks. We compute the log-likelihoods with vLLM (Kwon et al., 2023) from the log-probabilities of the input tokens. Counts, means, and differences of $\Delta\mathrm{NLL}_{\mathrm{baseline}}$ reported in the tables are computed from unrounded values.

D Accuracy Evaluation Details

This section details the accuracy evaluation of the 2B-scale models in § 5.1. We use accuracy at this scale, instead of the task loss used in § 4, because accuracy becomes more stable as models grow. At the 570M scale, accuracy on several downstream tasks, such as ARC-Challenge and OpenBookQA, stays within a few points of chance, so differences between closely matched configurations are hard to distinguish from evaluation noise. At the 2B scale, the baseline exceeds chance by a clear margin on most multiple-choice tasks (Table 3), which makes accuracy differences between models easier to interpret.

Evaluation Tasks We use the LM Evaluation Harness (Gao et al., 2024) for this evaluation. The evaluation set consists of the 7 tasks used for the task-loss evaluation in § 4 (the *selection set*) and 5 tasks not used for configuration selection (the *held-out set*). The held-out set comprises LAMBADA (Paperno et al., 2016), RACE (Lai et al., 2017), SciQ (Welbl et al., 2017), Social IQa (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2021), and serves to test whether the gains are specific to the tasks used for configuration selection. We evaluate RACE in a 0-shot setting, SciQ in a 3-shot setting, and the remaining tasks in a 5-shot setting. We report exact match for Natural Questions and TriviaQA and accuracy for the other tasks.

Checkpoint Averaging Because downstream-task accuracy fluctuates across checkpoints (Nishida et al., 2025), evaluation at a single checkpoint is unstable. We therefore evaluate a model whose weights are the average of the 5 checkpoints taken every $1{,}000$ steps over the final 20% of training (steps $21{,}000$–$25{,}000$). In contrast, the 570M-scale evaluation uses the final checkpoint (Appendix C).

Significance Testing We assess the significance of differences from the baseline with a paired bootstrap. For each task, we resample the evaluation examples with replacement $10{,}000$ times and, in each replicate, compute the accuracy difference between the two models on the same resampled examples. In each replicate, the difference in average accuracy is the mean of the per-task differences. We regard a difference as significant when its 95% percentile interval excludes zero. Because each model is pretrained only once, this interval reflects the uncertainty from sampling the evaluation examples for these two fixed models and does not include the variability across pretraining runs with different random seeds. A significant difference therefore indicates that the advantage is unlikely to be an artifact of the particular evaluation examples, whereas whether it persists across pretraining seeds remains untested (§ 6). All differences in Table 3 are computed before rounding.

E Theoretical Training and Inference FLOPs

Table 6 lists the theoretical FLOPs of all models used in our experiments, together with their FFN intermediate dimensions and non-embedding parameter counts.

Counting Conventions We count only matrix multiplications, with a multiply–add counted as two FLOPs; embedding lookups, normalization, element-wise operations (including the gating product in SwiGLU), softmax, and residual additions are excluded because their cost is negligible. A shared FFN is counted once for every layer that applies it, because parameter sharing reduces the number of parameters but not the computation. Let V denote the vocabulary size, S the sequence length, and d_{FF} the FFN intermediate dimension ($d_{\mathrm{FF}}^{\mathrm{base}}$ for the baselines and $d_{\mathrm{FF}}^{\mathrm{share}}$ for the Sandwich-style FFN-shared models). Because all our models use standard multi-head attention whose total head dimension equals d, the weight matrices applied to each token, including the output head, contain $P = L(4d^2 + 3dd_{\mathrm{FF}}) + Vd$ parameters, where a shared FFN is counted once per layer. In training, the forward and backward passes through a weight matrix with p parameters cost $2p$ and $4p$ FLOPs per token, respectively. With causal masking, the attention score and value products cost $2LSd$ FLOPs per token on average in the forward pass, and we count 2.5 times this cost for the backward pass, in which the attention probabilities are recomputed. The training FLOPs per token are therefore $6P + 7LSd$, with $S = 1{,}024$ and without activation checkpointing. In inference, prefill processes one sequence of $S = 1{,}024$ tokens and applies the output head only to its last token, which costs $2SL(4d^2 + 3dd_{\mathrm{FF}}) + 2Vd + 2LS^2d$ FLOPs. Decode generates one token attending to a KV cache of $C = 1{,}024$ tokens, which costs $2P + 4LCd$ FLOPs.

Dependence on N Because every layer applies an FFN of width $d_{\mathrm{FF}}^{\mathrm{share}}$, which is determined by N (§ 3.2), the FLOPs of a Sandwich-style FFN-shared model depend only on N and not on where the layer-specific FFNs are placed; for example, mirrored configurations (a, b) and (b, a) have identical FLOPs. As N decreases, the FLOPs increase, whereas the number of non-embedding parameters remains nearly constant. The selected $(n_{\mathrm{early}}, n_{\mathrm{late}}) = (1, 5)$ configuration ($N = 7$) requires $2.30\times$ the training FLOPs of the baseline at the 570M scale and $3.95\times$ at the 2B scale; the ratio is larger at the 2B scale because $d_{\mathrm{FF}}^{\mathrm{share}}/d_{\mathrm{FF}}^{\mathrm{base}} \approx L/N$ grows with L for a fixed N.

F 570M Baseline Model Per-task NLL

Because the main results report relative NLL differences to the baseline, Table 7 reports the absolute per-token NLL values of the 570M baseline model used as the reference in § 4.

G Ablation on FFN Expansion

To isolate the role of FFN expansion, we pretrain a model that keeps the selected $(n_{\mathrm{early}}, n_{\mathrm{late}}) = (1, 5)$ sharing pattern but fixes the FFN intermediate dimension to the baseline width $d_{\mathrm{FF}}^{\mathrm{base}}$. Every layer then computes an FFN of the same width as the baseline, so this model matches the baseline exactly in FLOPs, both in training and in inference (Table 6), while its non-embedding parameter count drops from 570.2M to 277.7M, a reduction of about 51%. As Table 8 shows, this no-expansion model underperforms the baseline by 0.097 on validation loss and by 0.029 on average task loss. This suggests that FFN expansion can enable a Sandwich-style model with shared FFNs to outperform the baseline. The table separates the cost of sharing from the gain of expansion. The baseline and the no-expansion $(1, 5)$ model have the same FFN width and the same FLOPs and differ only in whether the middle layers share FFN parameters, so this difference (0.029 on average task loss and 0.097 on validation loss) can be read as the cost of sharing. The no-expansion $(1, 5)$ model and $(1, 5)$ have the same sharing pattern and differ only in FFN width, so this difference (0.250 on average task loss) can be read as the gain from expansion. The difference between $(1, 5)$ and the baseline can therefore be read as the difference between this gain and this cost. We also note that the no-expansion model halves the non-embedding parameter count while matching the baseline in FLOPs (Table 6). Although it does not match the baseline in performance, its average task loss is only 0.029 worse, which may make it a useful option when memory, rather than computation, is the limiting resource.

Table 6: Theoretical training and inference FLOPs of the 570M-scale and 2B-scale models (see Appendix E for the counting conventions). N is the number of independent FFN parameter sets ($N = L$ for the baselines), d_{FF} is the FFN intermediate dimension, and # Params is the number of non-embedding parameters, in which shared FFN parameters are counted once. Parentheses show ratios to the baseline of the same scale. †The $(1, 5)$ configuration without FFN expansion (Appendix G).

Model	N	d_{FF}	# Params (M)	Training (GFLOPs/token)	Prefill (TFLOPs/seq.)	Decode (GFLOPs/token)
570M scale ($L = 24$)						
Baseline	24	4,480	570.2	4.63 (1.00×)	1.23 (1.00×)	1.59 (1.00×)
Sandwich-style	1	107,552	570.3	61.62 (13.32×)	20.69 (16.79×)	20.59 (12.91×)
	2	53,792	570.4	31.89 (6.89×)	10.54 (8.55×)	10.68 (6.70×)
	3	35,872	570.6	21.98 (4.75×)	7.16 (5.81×)	7.38 (4.63×)
	4	26,912	570.7	17.03 (3.68×)	5.47 (4.44×)	5.73 (3.59×)
	5	21,536	570.8	14.06 (3.04×)	4.45 (3.61×)	4.74 (2.97×)
	6	17,952	570.9	12.08 (2.61×)	3.78 (3.06×)	4.08 (2.56×)
	7	15,360	570.2	10.64 (2.30×)	3.29 (2.67×)	3.60 (2.26×)
	8	13,440	570.2	9.58 (2.07×)	2.92 (2.37×)	3.25 (2.04×)
	9	11,968	570.9	8.77 (1.90×)	2.65 (2.15×)	2.97 (1.87×)
	10	10,752	570.2	8.09 (1.75×)	2.42 (1.96×)	2.75 (1.73×)
	11	9,792	570.9	7.56 (1.63×)	2.24 (1.81×)	2.57 (1.61×)
	12	8,960	570.2	7.10 (1.54×)	2.08 (1.69×)	2.42 (1.52×)
	13	8,288	571.1	6.73 (1.46×)	1.95 (1.58×)	2.30 (1.44×)
	14	7,680	570.2	6.40 (1.38×)	1.84 (1.49×)	2.18 (1.37×)
	15	7,168	570.2	6.11 (1.32×)	1.74 (1.41×)	2.09 (1.31×)
	16	6,720	570.2	5.86 (1.27×)	1.66 (1.34×)	2.01 (1.26×)
	17	6,336	571.0	5.65 (1.22×)	1.58 (1.28×)	1.94 (1.21×)
	18	5,984	571.0	5.46 (1.18×)	1.52 (1.23×)	1.87 (1.17×)
	19	5,664	570.6	5.28 (1.14×)	1.46 (1.18×)	1.81 (1.14×)
	20	5,376	570.2	5.12 (1.11×)	1.40 (1.14×)	1.76 (1.10×)
	21	5,120	570.2	4.98 (1.08×)	1.35 (1.10×)	1.71 (1.07×)
	22	4,896	571.0	4.86 (1.05×)	1.31 (1.06×)	1.67 (1.05×)
	23	4,704	572.8	4.75 (1.03×)	1.27 (1.03×)	1.64 (1.03×)
No FFN expansion†	7	4,480	277.7	4.63 (1.00×)	1.23 (1.00×)	1.59 (1.00×)
2B scale ($L = 40$)						
Baseline	40	6,720	2,138.3	14.86 (1.00×)	4.54 (1.00×)	5.08 (1.00×)
Sandwich-style $(1, 5)$	7	38,432	2,139.5	58.70 (3.95×)	19.50 (4.30×)	19.70 (3.87×)

Table 7: Per-token NLL of the 570M baseline model used as the reference for $\Delta\mathrm{NLL}_{\mathrm{baseline}}$ in § 4. Values are reported in nats.

Metric	Baseline NLL
Validation loss	2.680
Average task loss	3.996
Natural Questions	4.249
HellaSwag	2.749
ARC-Easy	3.102
ARC-Challenge	3.605
OpenBookQA	5.363
PIQA	3.537
TriviaQA	5.367

H Top-20 Configurations for All Metrics

Table 9 lists the top-20 configurations for each of the nine metrics. The configuration $(1, 5)$ examined in § 5 is not an isolated optimum. Among the 9 configurations in its 3×3 neighborhood ($n_{\mathrm{early}} \in \{0, 1, 2\}$ and $n_{\mathrm{late}} \in \{4, 5, 6\}$), 6 rank among the top-20 on average task loss, and 3 to 6 do so on

Table 8: Effect of FFN expansion for the $(n_{\text{early}}, n_{\text{late}}) = (1, 5)$ sharing pattern. We report $\Delta\text{NLL}_{\text{baseline}} = \text{NLL}_{\text{baseline}} - \text{NLL}_{\text{target}}$ in nats; positive values indicate improvements over the 570M baseline, which has 570.2M non-embedding parameters. The last column reports the difference between the two models, computed before rounding.

	$(1, 5)$	$(1, 5)$ w/o expansion	Difference
Non-embedding parameters	570.2M	277.7M	
Training FLOPs ($\times$ baseline)	2.30	1.00	
Validation loss	-0.011	-0.097	0.087
Average task loss	0.221	-0.029	0.250
Natural Questions	0.192	-0.052	0.244
HellaSwag	0.026	-0.063	0.089
ARC-Easy	0.471	0.067	0.404
ARC-Challenge	0.423	0.129	0.293
OpenBookQA	0.137	-0.158	0.295
PIQA	0.146	-0.002	0.147
TriviaQA	0.154	-0.123	0.277

Table 9: Top-20 Sandwich-style FFN-shared configurations $(n_{\text{early}}, n_{\text{late}})$ ranked by $\Delta\text{NLL}_{\text{baseline}}$ for each of the 9 metrics, listed left-to-right in descending order of improvement. The configuration $(1, 5)$ examined in § 5 and its 8-neighbors are shown in **red** and **blue**, respectively.

Metric	Top-20 configurations $(n_{\text{early}}, n_{\text{late}})$
Val. Loss	(0, 2), **(0,4)**, (0, 1), (1, 3), (0, 3), **(0,5)**, (0, 0), (2, 2), (1, 2), **(1,4)**, **(0,6)**, (4, 4), (0, 8), (1, 1), **(2,6)**, (2, 3), (0, 12), (0, 16), (12, 8), (3, 4)
Avg. Loss	**(1,5)**, (0, 2), (0, 0), (4, 5), **(0,5)**, **(2,4)**, (5, 8), (8, 10), (5, 6), (1, 8), (0, 14), (2, 10), (6, 5), **(2,6)**, **(1,6)**, (2, 16), (1, 10), **(1,4)**, (2, 18), (2, 14)
HS	(0, 2), (0, 1), (0, 0), **(0,4)**, **(0,5)**, (0, 8), (0, 3), (0, 14), (1, 3), (1, 10), (2, 3), (1, 8), **(0,6)**, (0, 12), **(2,6)**, (1, 18), (5, 8), (2, 12), (2, 8), **(1,5)**
NQ	(0, 1), (0, 3), **(1,5)**, (0, 2), **(0,5)**, **(0,4)**, **(1,4)**, **(2,4)**, (1, 1), (1, 16), (3, 2), (1, 8), **(0,6)**, (0, 0), (2, 2), (1, 10), (2, 10), (6, 2), (8, 10), (0, 16)
ARC-E	**(1,5)**, (4, 5), **(2,4)**, (0, 2), (5, 8), (8, 10), (0, 0), (6, 5), (2, 16), (5, 6), (0, 14), (2, 12), **(1,6)**, (3, 12), (5, 12), (3, 18), (8, 4), (12, 8), (2, 10), **(2,6)**
ARC-C	(4, 5), **(1,5)**, **(2,4)**, (5, 8), (2, 16), (5, 6), (0, 0), (0, 2), (8, 10), (12, 8), (2, 12), (2, 10), (0, 14), (6, 5), (3, 12), (1, 10), **(1,6)**, (5, 3), **(0,5)**, (2, 18)
OBQA	**(0,5)**, (0, 20), (4, 14), (1, 8), (1, 12), (5, 8), (6, 16), (6, 4), (2, 18), (0, 0), (2, 3), (8, 2), (5, 14), **(1,5)**, **(2,6)**, (14, 5), **(1,6)**, (2, 10), (1, 0), (1, 18)
PIQA	(0, 0), (0, 14), **(0,5)**, (4, 5), (12, 8), **(1,5)**, (0, 2), (3, 8), (18, 2), (1, 10), (0, 12), (5, 6), (3, 18), (1, 12), **(0,4)**, (3, 12), (1, 3), (4, 14), (8, 10), (2, 3)
TQA	**(1,6)**, (4, 16), (0, 2), **(1,5)**, **(2,6)**, (0, 0), (5, 6), (1, 1), (1, 8), (6, 5), (0, 16), (8, 10), (0, 14), (0, 1), (1, 18), (2, 10), (5, 14), (5, 8), (2, 14), (3, 12)

each of the seven downstream tasks. Moreover, $(1, 5)$ itself ranks among the top-20 on all seven downstream tasks. This consistent high performance across neighboring configurations suggests that the strong performance of $(1, 5)$ is part of a broader pattern in the sampled parameter allocation space, rather than an isolated favorable result.

I HEATMAPS FOR ALL METRICS

Figures 5–13 show the heatmaps of Figure 3 enlarged so that the cell values are legible, together with those of the five downstream tasks not shown there (ARC-Easy, ARC-Challenge, OpenBookQA, PIQA, and TriviaQA).

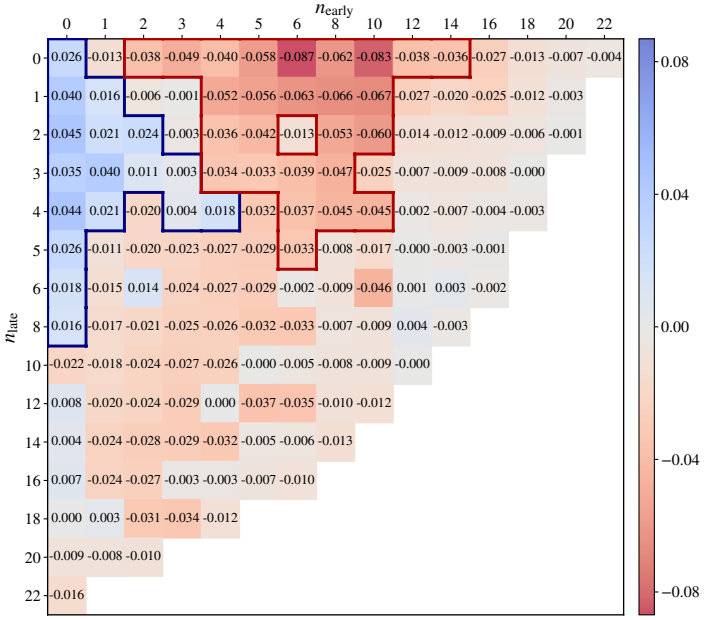


Figure 5: **Validation loss.** $\Delta\text{NLL}_{\text{baseline}}$ of validation loss for all 147 Sandwich-style FFN-shared models (570M scale), enlarged from Figure 3(a). The horizontal axis is n_{early} and the vertical axis is n_{late}. Each cell shows $\Delta\text{NLL}_{\text{baseline}}$ for the corresponding $(n_{\text{early}}, n_{\text{late}})$, and the blank region (lower right) corresponds to undefined configurations. Blue, red, and white indicate improvement, degradation, and parity over the baseline, respectively. The blue and red outlines mark the largest 4-connected component among the top-30 and bottom-30 cells of $\Delta\text{NLL}_{\text{baseline}}$, respectively.

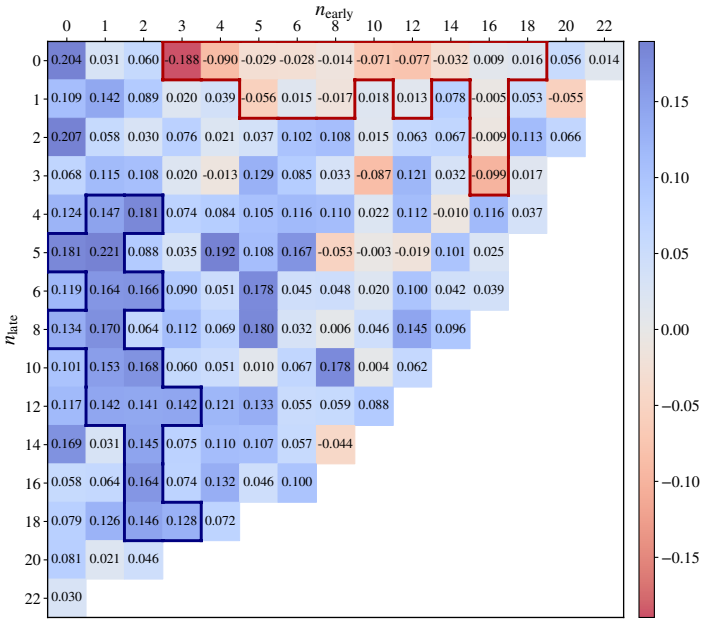


Figure 6: **Average task loss.** Same as Figure 5, but for task loss averaged over the seven downstream tasks, enlarged from Figure 3(b). All task-loss heatmaps (Figures 6–13) share one color scale, independent of that of validation loss; cells outside its range are drawn in the end colors of the color bar.

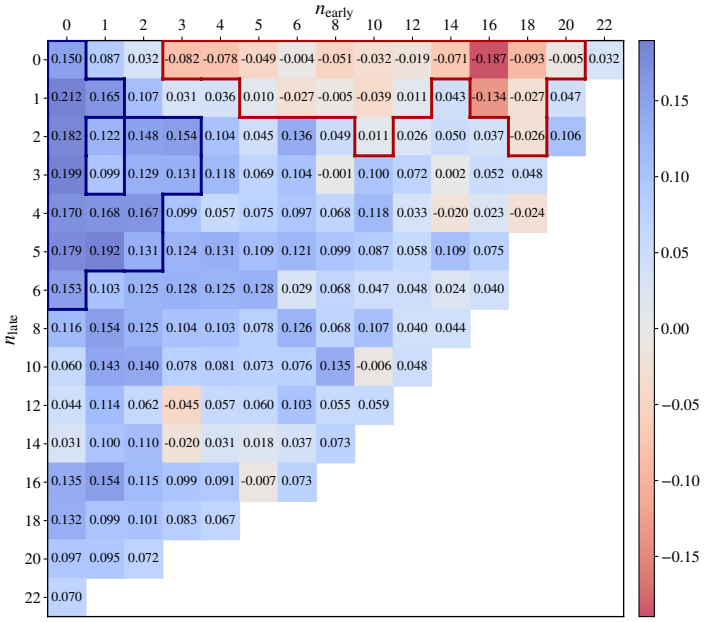


Figure 7: **Natural Questions.** Same as Figure 6, but for the task loss on Natural Questions, enlarged from Figure 3(c).

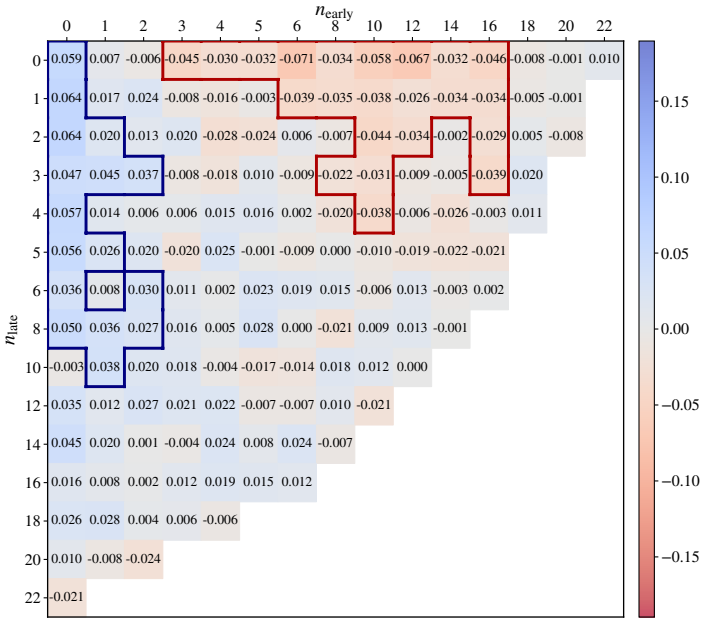


Figure 8: **HellaSwag.** Same as Figure 6, but for the task loss on HellaSwag, enlarged from Figure 3(d).

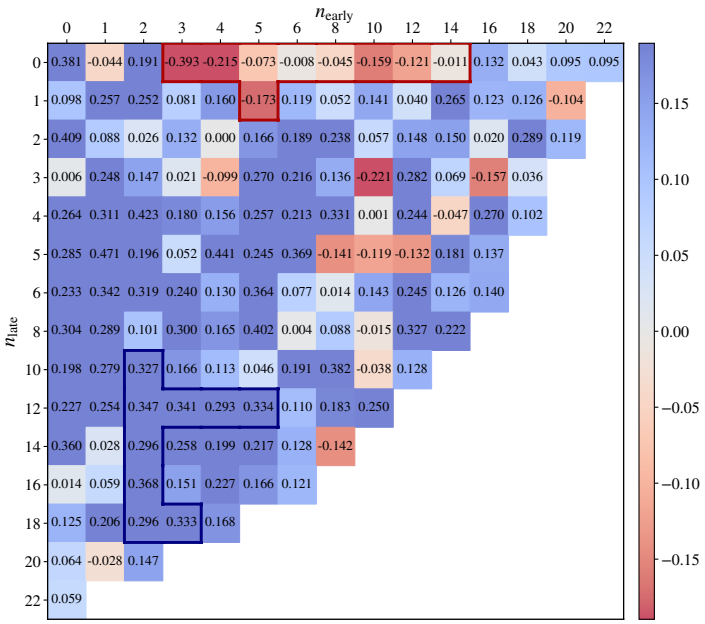


Figure 9: **ARC-Easy.** Same as Figure 6, but for the task loss on ARC-Easy.

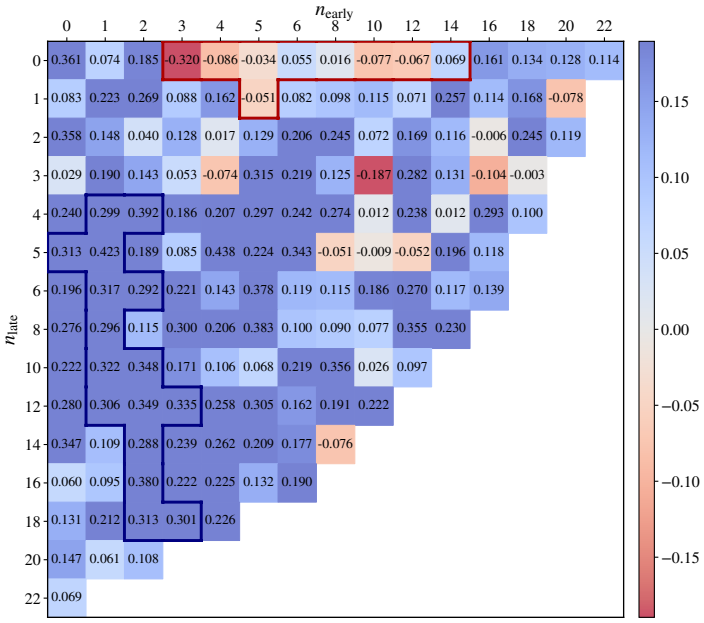


Figure 10: **ARC-Challenge.** Same as Figure 6, but for the task loss on ARC-Challenge.

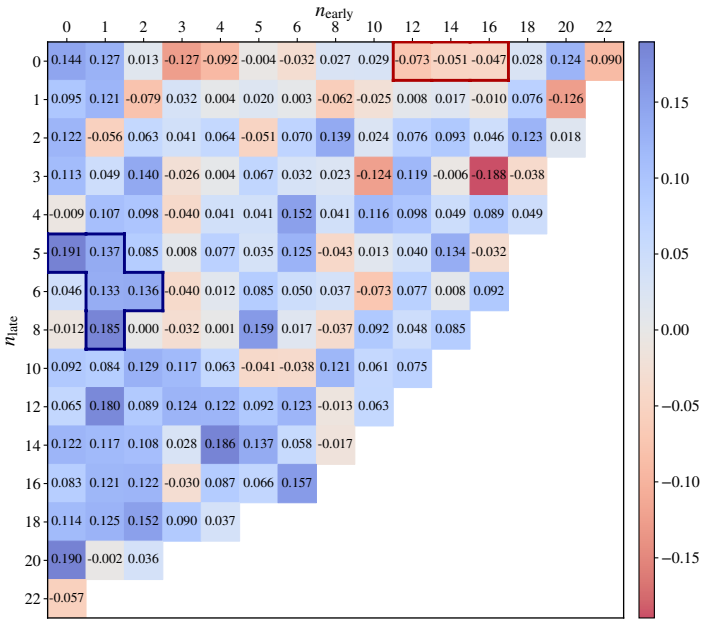


Figure 11: **OpenBookQA.** Same as Figure 6, but for the task loss on OpenBookQA.

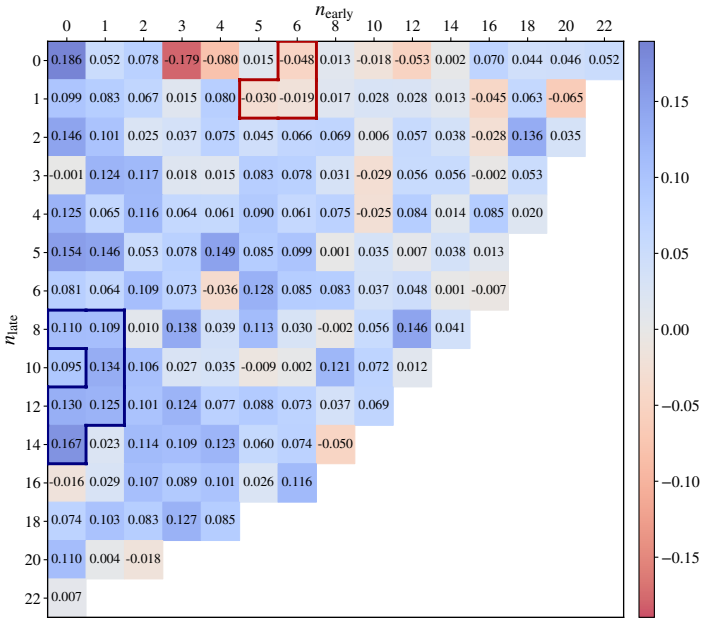


Figure 12: **PIQA.** Same as Figure 6, but for the task loss on PIQA.

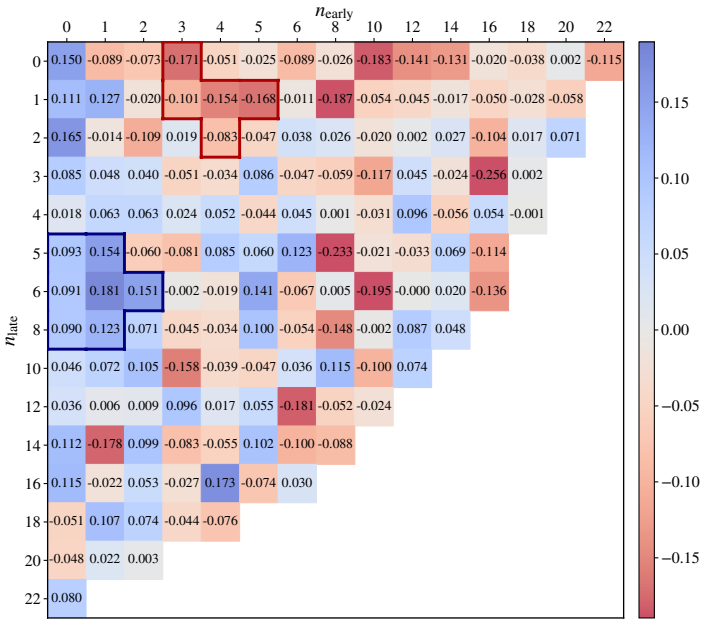


Figure 13: **TriviaQA.** Same as Figure 6, but for the task loss on TriviaQA.

Table 10: Results with a different pretraining corpus. The 570M baseline and the $(n_{\text{early}}, n_{\text{late}}) = (1, 5)$ Sandwich-style FFN-shared model are pretrained on WebOrganizer Corpus-200B (Wettig et al., 2025) instead of FineWeb-Edu, with all other settings identical to § 4. We report the per-token NLL of each model and $\Delta\text{NLL}_{\text{baseline}} = \text{NLL}_{\text{baseline}} - \text{NLL}_{\text{target}}$, all in nats; positive values indicate improvements over the baseline. Validation loss is computed on samples held out from WebOrganizer Corpus-200B. Differences are computed before rounding.

	NLL		
Metric	Baseline	$(1, 5)$	$\Delta\text{NLL}_{\text{baseline}}$
Validation loss	2.763	2.727	0.036
Average task loss	3.984	3.872	0.112
Natural Questions	4.048	3.956	0.092
HellaSwag	2.692	2.654	0.038
ARC-Easy	3.521	3.276	0.245
ARC-Challenge	3.808	3.609	0.199
OpenBookQA	5.309	5.250	0.059
PIQA	3.299	3.226	0.074
TriviaQA	5.210	5.135	0.074

J Robustness to the Pretraining Corpus

All experiments in the main text use FineWeb-Edu as the pretraining corpus. To examine whether the advantage of Sandwich-style FFN sharing is specific to this corpus, we pretrain the 570M baseline and the $(n_{\text{early}}, n_{\text{late}}) = (1, 5)$ Sandwich-style FFN-shared model on WebOrganizer Corpus-200B (Wettig et al., 2025) instead. All other settings, including the model configurations, the number of training tokens, and the optimization hyperparameters, are identical to those in § 4. Validation loss is computed on samples held out from WebOrganizer Corpus-200B, and task loss is computed on the same seven downstream tasks as in the main experiments.

Table 10 shows that the Sandwich-style FFN-shared model outperforms the baseline on both validation loss ($\Delta\text{NLL}_{\text{baseline}} = 0.036$) and average task loss ($\Delta\text{NLL}_{\text{baseline}} = 0.112$), and it reduces task loss on all seven tasks. Although the improvement in average task loss is smaller than that on FineWeb-Edu (0.221), its direction is consistent across the two corpora. This suggests that the advantage of the $(1, 5)$ configuration is not an artifact of a particular pretraining corpus.

K LICENSES OF EXISTING ASSETS

This work uses the following existing assets. We thank their creators and respect the terms of their respective licenses.

Datasets

- **FineWeb-Edu** (Penedo et al., 2024) — used as the pretraining corpus in the main experiments. License: ODC-BY 1.0. Available at `https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu`.
- **WebOrganizer Corpus-200B** (Wettig et al., 2025) — used as the pretraining corpus in § J. License: not specified on the dataset card; the corpus is derived from the DCLM-Pool (Li et al., 2024), which is released under CC BY 4.0. Available at `https://huggingface.co/datasets/WebOrganizer/Corpus-200B`.
- **HellaSwag** (Zellers et al., 2019) — License: MIT.
- **PIQA** (Bisk et al., 2019) — License: Academic Free License v3.0.
- **ARC** (Clark et al., 2018) — License: CC-BY-SA 4.0.
- **OpenBookQA** (Mihaylov et al., 2018) — License: Apache 2.0.
- **Natural Questions** (Kwiatkowski et al., 2019) — License: CC-BY-SA 3.0.
- **TriviaQA** (Joshi et al., 2017) — License: Apache 2.0.
- **LAMBADA** (Paperno et al., 2016) — License: CC BY 4.0.
- **RACE** (Lai et al., 2017) — License: custom terms of use by the dataset authors; available for non-commercial research purposes only (`https://www.cs.cmu.edu/~glai1/data/race/`).
- **SciQ** (Welbl et al., 2017) — License: CC BY-NC 3.0.
- **Social IQa** (Sap et al., 2019) — License: CC BY 4.0.
- **WinoGrande** (Sakaguchi et al., 2021) — License: CC-BY.

Tokenizer

- **Llama 3.1 tokenizer** (Grattafiori et al., 2024) — governed by the Llama 3.1 Community License Agreement.

Software

- **Meta Lingua** (Videau et al., 2024) — pretraining framework. License: BSD-3-Clause. Available at `https://github.com/facebookresearch/lingua`.
- **LM Evaluation Harness** (Gao et al., 2024) — License: MIT. Available at `https://github.com/EleutherAI/lm-evaluation-harness`.
- **vLLM** (Kwon et al., 2023) — inference engine for downstream task evaluation. License: Apache 2.0. Available at `https://github.com/vllm-project/vllm`.
- **Hugging Face Transformers** (Wolf et al., 2020) — License: Apache 2.0. Available at `https://github.com/huggingface/transformers`. Our model implementation is derived from the Llama implementation in Hugging Face Transformers, with modifications to support Sandwich-style FFN sharing.

All assets were used in accordance with their respective licenses, and only for non-commercial research purposes.

Table 11: Model configurations of all Sandwich-style FFN-shared variants examined in Section 4, built on a 570M-parameter, 24-layer baseline. Each row specifies the number of early and late unshared layers (n_{early}, n_{late}), the shared layer indices, the FFN intermediate dimension d_{FF}, which is common to all layers and is set to $d_{\text{FF}}^{\text{share}}$ from the equation in § 3.2 rounded up to a multiple of 32 (or to $d_{\text{FF}}^{\text{base}}$ for the baseline), and the total parameter counts with and without the embedding. The last row ($N = 24$, no shared layers) corresponds to the baseline model.

N	n_{early}	n_{late}	Shared layers	d_{FF}	# Params (w/o emb.)	# Params (w/ emb.)
1	0	0	1–24	107,552	570,319,360	898,654,720
2	1	0	2–24	53,792	570,443,520	898,778,880
3	2	0	3–24	35,872	570,567,680	898,903,040
4	3	0	4–24	26,912	570,691,840	899,027,200
5	4	0	5–24	21,536	570,816,000	899,151,360
6	5	0	6–24	17,952	570,940,160	899,275,520
7	6	0	7–24	15,360	570,204,160	898,539,520
9	8	0	9–24	11,968	570,944,000	899,279,360
11	10	0	11–24	9,792	570,946,560	899,281,920
13	12	0	13–24	8,288	571,072,000	899,407,360
15	14	0	15–24	7,168	570,214,400	898,549,760
17	16	0	17–24	6,336	570,954,240	899,289,600
19	18	0	19–24	5,664	570,588,160	898,923,520
21	20	0	21–24	5,120	570,222,080	898,557,440
23	22	0	23–24	4,704	572,805,120	901,140,480
2	0	1	1–23	53,792	570,443,520	898,778,880
3	1	1	2–23	35,872	570,567,680	898,903,040
4	2	1	3–23	26,912	570,691,840	899,027,200
5	3	1	4–23	21,536	570,816,000	899,151,360
6	4	1	5–23	17,952	570,940,160	899,275,520
7	5	1	6–23	15,360	570,204,160	898,539,520
8	6	1	7–23	13,440	570,205,440	898,540,800
10	8	1	9–23	10,752	570,208,000	898,543,360
12	10	1	11–23	8,960	570,210,560	898,545,920
14	12	1	13–23	7,680	570,213,120	898,548,480
16	14	1	15–23	6,720	570,215,680	898,551,040
18	16	1	17–23	5,984	570,955,520	899,290,880
20	18	1	19–23	5,376	570,220,800	898,556,160
22	20	1	21–23	4,896	570,960,640	899,296,000
3	0	2	1–22	35,872	570,567,680	898,903,040
4	1	2	2–22	26,912	570,691,840	899,027,200
5	2	2	3–22	21,536	570,816,000	899,151,360
6	3	2	4–22	17,952	570,940,160	899,275,520
7	4	2	5–22	15,360	570,204,160	898,539,520
8	5	2	6–22	13,440	570,205,440	898,540,800
9	6	2	7–22	11,968	570,944,000	899,279,360
11	8	2	9–22	9,792	570,946,560	899,281,920
13	10	2	11–22	8,288	571,072,000	899,407,360
15	12	2	13–22	7,168	570,214,400	898,549,760
17	14	2	15–22	6,336	570,954,240	899,289,600
19	16	2	17–22	5,664	570,588,160	898,923,520
21	18	2	19–22	5,120	570,222,080	898,557,440
23	20	2	21–22	4,704	572,805,120	901,140,480
4	0	3	1–21	26,912	570,691,840	899,027,200
5	1	3	2–21	21,536	570,816,000	899,151,360
6	2	3	3–21	17,952	570,940,160	899,275,520
7	3	3	4–21	15,360	570,204,160	898,539,520
8	4	3	5–21	13,440	570,205,440	898,540,800
9	5	3	6–21	11,968	570,944,000	899,279,360

(continued on next page)

(continued from previous page)

N	n_{early}	n_{late}	Shared layers	d_{FF}	# Params (w/o emb.)	# Params (w/ emb.)
10	6	3	7–21	10,752	570,208,000	898,543,360
12	8	3	9–21	8,960	570,210,560	898,545,920
14	10	3	11–21	7,680	570,213,120	898,548,480
16	12	3	13–21	6,720	570,215,680	898,551,040
18	14	3	15–21	5,984	570,955,520	899,290,880
20	16	3	17–21	5,376	570,220,800	898,556,160
22	18	3	19–21	4,896	570,960,640	899,296,000
5	0	4	1–20	21,536	570,816,000	899,151,360
6	1	4	2–20	17,952	570,940,160	899,275,520
7	2	4	3–20	15,360	570,204,160	898,539,520
8	3	4	4–20	13,440	570,205,440	898,540,800
9	4	4	5–20	11,968	570,944,000	899,279,360
10	5	4	6–20	10,752	570,208,000	898,543,360
11	6	4	7–20	9,792	570,946,560	899,281,920
13	8	4	9–20	8,288	571,072,000	899,407,360
15	10	4	11–20	7,168	570,214,400	898,549,760
17	12	4	13–20	6,336	570,954,240	899,289,600
19	14	4	15–20	5,664	570,588,160	898,923,520
21	16	4	17–20	5,120	570,222,080	898,557,440
23	18	4	19–20	4,704	572,805,120	901,140,480
6	0	5	1–19	17,952	570,940,160	899,275,520
7	1	5	2–19	15,360	570,204,160	898,539,520
8	2	5	3–19	13,440	570,205,440	898,540,800
9	3	5	4–19	11,968	570,944,000	899,279,360
10	4	5	5–19	10,752	570,208,000	898,543,360
11	5	5	6–19	9,792	570,946,560	899,281,920
12	6	5	7–19	8,960	570,210,560	898,545,920
14	8	5	9–19	7,680	570,213,120	898,548,480
16	10	5	11–19	6,720	570,215,680	898,551,040
18	12	5	13–19	5,984	570,955,520	899,290,880
20	14	5	15–19	5,376	570,220,800	898,556,160
22	16	5	17–19	4,896	570,960,640	899,296,000
7	0	6	1–18	15,360	570,204,160	898,539,520
8	1	6	2–18	13,440	570,205,440	898,540,800
9	2	6	3–18	11,968	570,944,000	899,279,360
10	3	6	4–18	10,752	570,208,000	898,543,360
11	4	6	5–18	9,792	570,946,560	899,281,920
12	5	6	6–18	8,960	570,210,560	898,545,920
13	6	6	7–18	8,288	571,072,000	899,407,360
15	8	6	9–18	7,168	570,214,400	898,549,760
17	10	6	11–18	6,336	570,954,240	899,289,600
19	12	6	13–18	5,664	570,588,160	898,923,520
21	14	6	15–18	5,120	570,222,080	898,557,440
23	16	6	17–18	4,704	572,805,120	901,140,480
9	0	8	1–16	11,968	570,944,000	899,279,360
10	1	8	2–16	10,752	570,208,000	898,543,360
11	2	8	3–16	9,792	570,946,560	899,281,920
12	3	8	4–16	8,960	570,210,560	898,545,920
13	4	8	5–16	8,288	571,072,000	899,407,360
14	5	8	6–16	7,680	570,213,120	898,548,480
15	6	8	7–16	7,168	570,214,400	898,549,760
17	8	8	9–16	6,336	570,954,240	899,289,600
19	10	8	11–16	5,664	570,588,160	898,923,520
21	12	8	13–16	5,120	570,222,080	898,557,440
23	14	8	15–16	4,704	572,805,120	901,140,480

(continued on next page)

(continued from previous page)

N	n_{early}	n_{late}	Shared layers	d_{FF}	# Params (w/o emb.)	# Params (w/ emb.)
11	0	10	1–14	9,792	570,946,560	899,281,920
12	1	10	2–14	8,960	570,210,560	898,545,920
13	2	10	3–14	8,288	571,072,000	899,407,360
14	3	10	4–14	7,680	570,213,120	898,548,480
15	4	10	5–14	7,168	570,214,400	898,549,760
16	5	10	6–14	6,720	570,215,680	898,551,040
17	6	10	7–14	6,336	570,954,240	899,289,600
19	8	10	9–14	5,664	570,588,160	898,923,520
21	10	10	11–14	5,120	570,222,080	898,557,440
23	12	10	13–14	4,704	572,805,120	901,140,480
13	0	12	1–12	8,288	571,072,000	899,407,360
14	1	12	2–12	7,680	570,213,120	898,548,480
15	2	12	3–12	7,168	570,214,400	898,549,760
16	3	12	4–12	6,720	570,215,680	898,551,040
17	4	12	5–12	6,336	570,954,240	899,289,600
18	5	12	6–12	5,984	570,955,520	899,290,880
19	6	12	7–12	5,664	570,588,160	898,923,520
21	8	12	9–12	5,120	570,222,080	898,557,440
23	10	12	11–12	4,704	572,805,120	901,140,480
15	0	14	1–10	7,168	570,214,400	898,549,760
16	1	14	2–10	6,720	570,215,680	898,551,040
17	2	14	3–10	6,336	570,954,240	899,289,600
18	3	14	4–10	5,984	570,955,520	899,290,880
19	4	14	5–10	5,664	570,588,160	898,923,520
20	5	14	6–10	5,376	570,220,800	898,556,160
21	6	14	7–10	5,120	570,222,080	898,557,440
23	8	14	9–10	4,704	572,805,120	901,140,480
17	0	16	1–8	6,336	570,954,240	899,289,600
18	1	16	2–8	5,984	570,955,520	899,290,880
19	2	16	3–8	5,664	570,588,160	898,923,520
20	3	16	4–8	5,376	570,220,800	898,556,160
21	4	16	5–8	5,120	570,222,080	898,557,440
22	5	16	6–8	4,896	570,960,640	899,296,000
23	6	16	7–8	4,704	572,805,120	901,140,480
19	0	18	1–6	5,664	570,588,160	898,923,520
20	1	18	2–6	5,376	570,220,800	898,556,160
21	2	18	3–6	5,120	570,222,080	898,557,440
22	3	18	4–6	4,896	570,960,640	899,296,000
23	4	18	5–6	4,704	572,805,120	901,140,480
21	0	20	1–4	5,120	570,222,080	898,557,440
22	1	20	2–4	4,896	570,960,640	899,296,000
23	2	20	3–4	4,704	572,805,120	901,140,480
23	0	22	1–2	4,704	572,805,120	901,140,480
24	–	–	–	4,480	570,225,920	898,561,280

Table 12: Model configurations of the Sandwich-style FFN-shared variant and the baseline examined in § 5.1, built on a 2B-parameter, 40-layer model. Each row specifies the number of early and late unshared layers (n_{early}, n_{late}), the shared layer indices, the FFN intermediate dimension d_{FF}, which is common to all layers and is set to $d_{\text{FF}}^{\text{share}}$ from the equation in § 3.2 rounded up to a multiple of 32 (or to $d_{\text{FF}}^{\text{base}}$ for the baseline), and the total parameter counts with and without the embedding. The last row ($N = 40$, no shared layers) corresponds to the baseline model.

N	n_{early}	n_{late}	Shared layers	d_{FF}	# Params (w/o emb.)	# Params (w/ emb.)
7	1	5	2–35	38,432	2,139,494,400	2,631,997,440
40	–	–	–	6,720	2,138,267,520	2,630,770,560