DO DEPTH-GROWN MODELS OVERCOME THE CURSE OF DEPTH? AN IN-DEPTH ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradually growing the depth of Transformers during training cannot only reduce training cost but also lead to improved reasoning performance, as shown by MIDAS [Saunshi et al., 2024]. Thus far, however, a mechanistic understanding of these gains has been missing. In this work, we establish a connection to recent work showing that layers in the second half of non-grown, pre-layernorm Transformers contribute much less to the final output distribution than those in the first half—also known as the *Curse of Depth* [Sun et al., 2025b; Csordás et al., 2025]. Using depth-wise analyses, we show that growth via gradual middle stacking yields more effective utilization of model depth, changes in the residual stream structure, and formation of permutable computational blocks. In addition, we propose a lightweight modification of MIDAS that yields further improvements in downstream reasoning benchmarks. Overall, this work highlights how gradual growth of model depth can lead to formation of distinct computational circuits and overcome the limited depth utilization seen in standard non-grown models.

1 INTRODUCTION

The remarkable success of large language models (LLMs) has been accompanied by immense computational and energy demands. This trend of training larger and larger networks is correlated with the increasing depth of model architectures [Kaplan et al., 2020; Hoffmann et al., 2022]. As Transformers [Vaswani et al., 2017] lack recurrence, their computational capacity is directly linked to their depth. Greater depth enables more complex computations and improves capabilities like reasoning, compositional generalization and goal reaching [Petty et al., 2023; Lad et al., 2024; Wang et al., 2025]. However, this pursuit of greater scale uncovers a critical inefficiency, as training such models is extremely resource-intensive [Varoquaux et al., 2025].

A core issue of the current paradigm is the observation that not all layers contribute equally to the final model's performance [Yin et al., 2023; Gromov et al., 2024; Li et al., 2024; Men et al., 2024]. Csordás et al. [2025] and Sun et al. [2025b] demonstrated the phenomenon that deeper layers of modern pre-layer Transformers tend to be less effective than their earlier counterparts, with many layers in the second half of the model contributing minimally to the final output — also known as the *Curse of Depth* [Sun et al., 2025b]. This observation, which highlights a kind of over-parametrization, is supported by findings that various architectures are remarkably robust to perturbations like swapping or skipping intermediate layers without significant performance loss [Lad et al., 2024; Yin et al., 2023]. The Curse of Depth represents a major resource inefficiency in today's paradigm. As highlighted by Csordás et al. [2025], addressing these limitations is a pressing need for the community to avoid the waste of valuable resources and to develop more efficient architectures that can leverage deep layers effectively.

A promising solution lies in gradually grown architectures, which dynamically expand a model's depth or width during training. These novel training strategies, such as gradual stacking [Gong et al., 2019; Reddi et al., 2023], enable efficient training by using layers from a smaller model to initialize the next stage. Of particular interest is the `MIDAS` method [Saunshi et al., 2024], which gradually increases depth by inserting new layers into the middle of the model. `MIDAS` has been shown to not only speed up training but also to improve performance on reasoning-heavy benchmarks, suggesting that this growth procedure introduces a favourable inductive bias. However, a clear mechanistic understanding of these gains has so far been missing.

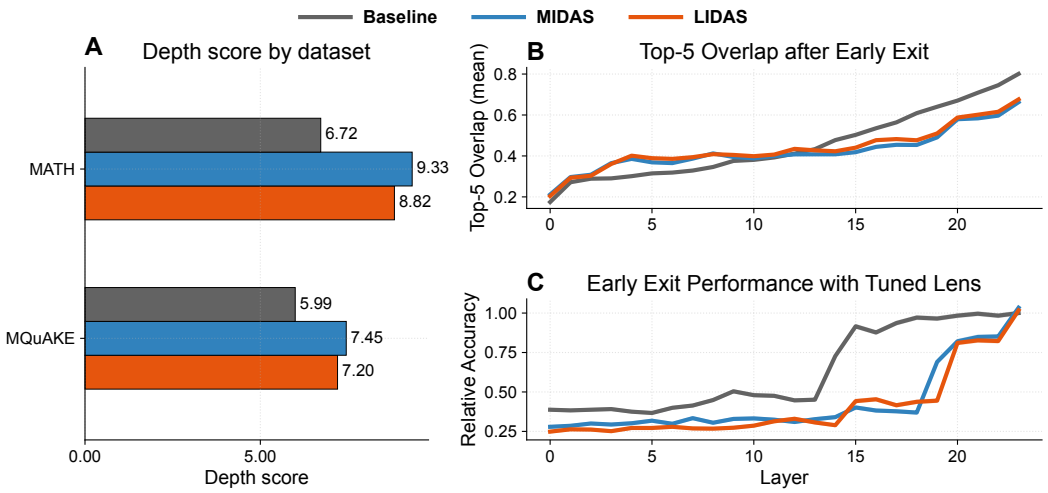


Figure 1: Depth-grown models use their depth more. (A) Depth score [Csordás et al., 2025] on MATH [Hendrycks et al., 2021] and MQuAKE [Zhong et al., 2023]. Grown models have consistently higher depth scores. (B) Top-5 overlap between each layer's early-exit vocabulary and model's final vocabulary on 20 prompts from GSM8K [Cobbe et al., 2021]. Both grown models studied in this work (`MIDAS`, `LIDAS`) show lower overlap at later layers, indicating that these later layers still add more features needed for the final prediction. (C) Early-exit relative accuracy versus layer (*Variable Assignment Math* primitive). The baseline reaches near its final performance early, whereas accuracy for `MIDAS` and `LIDAS` continues to rise up to the last layer.

In this work, we establish a direct connection between gradual depth growth and the "Curse of Depth" [Sun et al., 2025b], providing a mechanistic understanding of how gradual depth growth procedures can lead to a more effective utilization of a model's depth. Using depth analysis tools [Belrose et al., 2023; Csordás et al., 2025], we show that gradual stacking counteracts the patterns of diminishing returns observed in non-grown models. We summarize our contributions below:

- **`MIDAS` reproduction on different backbones.** We reproduce the core `MIDAS` results on SmolLM-v1 backbones (360M and 1.7B), trained with autoregressive next-token prediction, confirming that gradual depth growth improves reasoning performance over a conventionally trained, non-grown baseline with a 1.29x improvement in reduced training cost.
- **Extensive suite of analyses showing that grown models use their depth more.** We provide an in-depth analysis of how gradual depth growth alters computation and representation in LLMs, providing mechanistic insights into these findings. We show that grown models utilize their depth more efficiently than conventionally trained baselines. Furthermore, we demonstrate that grown models develop permutable computational blocks in the middle of the network, with each layer within a block fulfilling a specific cyclical role.
- **Novel gradual depth growth strategy `LIDAS`.** Based on our extensive analyses, we propose `LIDAS`, an improved growing strategy that duplicates the layer-wise middle rather than the block-wise middle while preserving the inductive bias of growing. Across scales, `LIDAS` matches or exceeds `MIDAS` and conventionally trained models in reasoning benchmarks without degrading Negative Log-Likelihood (NLL) or knowledge performance. In depth analyses, `LIDAS` produces more symmetric weights and even further utilizes its depth.

Overall, this work provides a first mechanistic understanding of how gradual depth growth can counteract the Curse of Depth, potentially leading to more efficient and capable language models.

2 Related Work

Growing Neural Networks. Early on, researchers recognized the advantages of training neural networks one layer at a time [Hinton et al., 2006; Bengio et al., 2006] to overcome the challenges

of learning long-term dependencies with gradient descent [Bengio et al., 1994]. More recently, this concept has been re-explored for large language models (LLMs) through gradual stacking [Gong et al., 2019; Reddi et al., 2023; Du et al., 2024] and depth up-scaling [Kim et al., 2023]. Alternative growing strategies include masked structural growth [Yao et al., 2023], function-preserving expansions [Gesmundo & Maile, 2023] and learned linear growth operators [Wang et al., 2023].

Adaptive Architectures. A potentially complementary strategy to growing is using adaptive architectures that dynamically adjust their computational graph or parameters based on their input data by using a larger, pre-trained network more efficiently, including mixture of experts approaches [Jacobs et al., 1991; Shazeer et al., 2017; Csordás et al., 2024] or early exiting [Teerapittayanon et al., 2016; Xin et al., 2020]. Recent approaches apply adaptive token level computations [Bae et al., 2025], nested models for elastic inference [Devvrit et al., 2024] or depth-wise looping [Giannou et al., 2023; Yang et al., 2023; von Oswald et al., 2025].

Depth of Neural Network Architectures. While depth is a key factor correlated with network performance [Csordás et al., 2025], recent research has found that deeper layers in LLMs are often redundant and less effective. This phenomenon has been termed the Curse of Depth, which suggests deeper layers contribute minimally to learning [Sun et al., 2025b]. Studies on models like GPT-2 show that their middle and deep layers exhibit remarkable robustness to significant perturbations, including layer swapping and deletion [Yin et al., 2023; Lad et al., 2024]. This over-provisioning has inspired various layer intervention strategies, such as skipping, swapping, or parallelization, to improve efficiency [Lad et al., 2024; Sun et al., 2025a].

Reasoning. For solving challenging tasks, recent work has shifted focus to recurrence and looping [Geiping et al., 2025; Saunshi et al., 2025] to improve model reasoning and leverage depth scaling for enhanced internal "thinking" [Chen et al., 2025]. These methods scale up test-time computation to allow models to iteratively refine their answers. Complementary to these approaches, our work focuses on identifying and leveraging computational blocks within depth-grown neural networks to improve reasoning, rather than relying on a fixed, recurrent process.

3 Two Depth-Grown Transformers: MIDAS & LIDAS

In this section, we first formalise the growth operator on a fixed architecture class $\mathcal{F}$ and recover `MIDAS` [Saunshi et al., 2024] as a special case. We then introduce `LIDAS`, which inserts a new middle block constructed by interleaving its neighbours to provide a stronger initialisation. Finally, using models from the SmolLM-v1 family [Ben Allal et al., 2024], we present empirical results on aggregated reasoning and knowledge benchmarks, showing that both gradual-depth growing methods outperform a conventionally trained, non-grown baseline on these tasks while remaining on par with general language-modelling performance.

3.1 The Growing operator

We fix a base architecture class $\mathcal{F}$ (width, heads, embedding size, etc. are fixed) and vary only depth. Let $f_L \in \mathcal{F}$ denote a model with L Transformer layers, written as an ordered list $f_L = [\ell_0, \ldots, \ell_{L-1}]$. A (depth) *growth operator* $G : \mathcal{F} \times \mathbb{N} \to \mathcal{F}$ maps an L-layer model to an $(L + b)$-layer model, such that $G(f_L; b) = f_{L+b}$, where $b \in \mathbb{N}$ is the *block size* (the number of layers added per growth step). Following Saunshi et al. [2024], we consider growth operators that insert new layers in the centre of the model and keep the block size b fixed across growing stages.

The following strategies use layer duplication to initialize new layers within the newly inserted block. This consists of deep copying all parameters within the layer, including their optimizer state. The result is two initially identical copies at different depths of the model, thus allowed to diverge as training continues.

`MIDAS`. When depth increases by b layers per stage, at stage n we have $L = nb$ and can partition f_L into n contiguous blocks of size b :

$$f_L = [B_0 \,\|\, B_1 \,\|\, \cdots \,\|\, B_{n-1}], \quad B_j = [\ell_{jb}, \ldots, \ell_{(j+1)b-1}]. \tag{1}$$

Let $m_b = \lceil \frac{n}{2} \rceil - 1$ denote the *middle block* index. Middle gradual stacking inserts a new block B' immediately after B_{m_b}, i.e.

$$G(f_L; b) = [B_0 \,\|\, \cdots \,\|\, B_{m_b} \,\|\, B' \,\|\, B_{m_b+1} \,\|\, \cdots \,\|\, B_{n-1}]. \tag{2}$$

If $B' = B_{m_b}$ (copying the middle block), we recover MIDAS as proposed in Saunshi et al. [2024].

LIDAS. Since we are constrained by the block patterning in MIDAS, we propose Layer-wise mIDdle grAdual Stacking, or LIDAS, in which we consider the *middle layer* $m_l = \lceil \frac{L}{2} \rceil$ to be the central point of the growing operation. We then construct a new block $B' = [l_{m_l - \lceil b/2 \rceil}, \ldots, l_{m_l + \lfloor b/2 \rfloor}]$, around the middle layer l_{m_l}, which is inserted after the layer $l_{m_l + \lfloor b/2 \rfloor}$. For odd-indexed growing stages, i.e., odd number of blocks, MIDAS and LIDAS coincide by selecting the same layers. They differ for even-indexed growing stages, shown from a block-wise perspective in Fig. 2. Further details can be found in Section A.

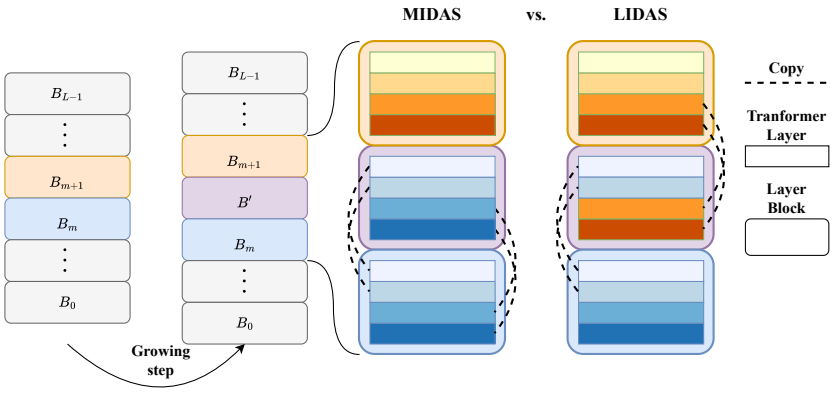


Figure 2: Illustration of growing strategies with block size 4: MIDAS vs. LIDAS, with an even number of existing blocks. MIDAS [Saunshi et al., 2024] simply copies $B' = B_m$, which is the block preceding mid-depth. When seen from a block-wise perspective instead of a layer-wise perspective, our proposed variant LIDAS may be interpreted as forming B' from the two blocks surrounding the mid-depth by combining the head of B_{m+1} with the tail of B_m in reverse order. This small difference in initialization leads to significantly improved performance as shown in Table 1.

Training runs and schedules. A training run is specified by (i) the model class $\mathcal{F}$, (ii) the target depth L_{final}, (iii) the initial depth L_0 (typically $L_0 = b$), (iv) a fixed block size b, and (v) a stage schedule $\{T_s\}_{s=0}^{S-1}$ (training steps per stage). Starting from f_{L_0}, after each stage s, we apply $G(\cdot; b)$ to obtain $f_{L_{s+1}}$ with depth $L_{s+1} = L_s + b$. We repeat until $L_S = L_{\text{final}}$.

3.2 Experiments

Setup. We evaluate and compare the two growing methods, MIDAS and LIDAS, against a conventionally non-grown baseline. We use the 360M and 1.7B models from the SmolLM-v1 family [Ben Allal et al., 2024] to probe scaling behaviour. All models are trained from scratch on the SmolLM-Corpus, a curated mixture of educational and synthetic texts as well as mathematics and code. Due to their favourable efficiency–performance trade-off, these models enable competitive evaluation within a constrained computational budget. For all grown models we present, we use the block size $b = 4$ and a prop-1 schedule (see Appendix Section A for details).

Benchmarks. We report negative log-likelihood (NLL) on a held-out validation set from the SmolLM-Corpus. We follow the knowledge and reasoning benchmarking suite reported in Saunshi et al. [2024]. The knowledge-based benchmarks are split into Open-book Q&A with provided context (TyDiQA-GoldP, SQuADv2, DROP, QuAC, CoQA), and Closed-book Q&A without context (TriviaQA, TyDiQA-NoContext, NaturalQuestions, WebQuestions), evaluated zero-shot. We additionally add Lambada [Paperno et al., 2016] and HellaSwag [Zellers et al., 2019] in their classical settings. For reasoning, we report the aggregated performance on MathWorld problems (SVAMP [Patel et al., 2021], ASDiv [Miao et al., 2021], AQuA and MAWPS [Koncel-Kedziorski et al., 2016]) and reasoning primitives, which are a suite of synthetic tasks designed by Saunshi et al. [2024] to specifically investigate reasoning performance on a smaller scale, both evaluated under five-shot prompting as done previously. For the exact score breakdown, we refer to Appendix Section C.

Results. Aggregated results are shown in Table 1. Consistent with the findings of Saunshi et al. [2024], we observe that depth-grown models (both MIDAS and LIDAS) outperform the

baseline on reasoning-heavy tasks (i.e., MathWorld and Reasoning Primitives). On the remaining benchmarks (Open-book Q&A, Closed-book Q&A, and Lambada), we observe little deviation from the baseline model with LIDAS being slightly superior to MIDAS. In addition, we observe a 1.29x improvement in reduced training cost for depth-grown models over a conventionally trained, non-grown baseline Table 7. In summary, our results reproduce the observation of Saunshi et al. [2024]. MIDAS outperforms a conventionally trained baseline on reasoning-heavy tasks, and our proposed method LIDAS further strengthens this effect without degrading NLL at 1.7B. To stabilise results on MathWorld, we additionally report the performance of models which are finetuned on the OpenWebMath dataset. While the relative order stays the same, we observe for the 360M model that the improvements for the grown models become more pronounced. However, the reasons behind these gains remain unclear. Therefore, we turn next to a detailed analysis of the 1.7B models, aiming to characterize how MIDAS and LIDAS may mechanistically differ from the baseline and how this could lead to improved performance in reasoning tasks.

		Standard cooldown							Math cooldown	
		Holdout Set (NLL ↓)	**Open-book Q&A** (F1 ↑)	**Closed-book Q&A** (F1 ↑)	**Lambada** (Acc ↑)	**Hellaswag** (Acc ↑)	**MathWorld** (Acc ↑)	**Primitives** (Acc ↑)	**MathWorld** (Acc ↑)	**Primitives** (Acc ↑)
360M	Baseline	2.18	22.90	**14.20**	43.35	39.97	3.69	30.06	8.10	33.12
	MIDAS	2.18	24.57	13.53	43.31	40.36	**4.39**	28.18	**13.43**	35.14
	LIDAS	**2.16**	**26.63**	14.08	**44.03**	**40.58**	4.36	**31.20**	12.30	**50.36**
1.7B	Baseline	**1.96**	29.57	18.61	50.05	46.28	13.75	34.86	23.28	42.77
	MIDAS	1.97	28.80	18.50	50.81	46.19	16.07	40.74	**24.60**	53.00
	LIDAS	**1.96**	**29.84**	**19.08**	**51.41**	**46.32**	**18.59**	**47.39**	24.01	**55.57**

Table 1: Performance comparison of a standard transformer baseline and the two grown models MIDAS and LIDAS. We reproduce the findings of Saunshi et al. [2024] and observe that grown models match the baseline in training objective (NLL), standard Q&A benchmarks as well as Lambada. Grown models, especially LIDAS, outperform the non-grown baseline on reasoning-heavy tasks such as MathWorld and Primitives.

4 Depth Analysis

Motivated by the confirmed observations that gradually depth-grown Transformers seem to yield increased reasoning abilities, we investigate here how gradual depth growth reshapes computation across depth in fully trained models. To this end, we first examine early-exiting performance for every layer with TunedLens [Belrose et al., 2023] to test how much performance degrades when we exit early. Next, we run various interventions on the models, such as swapping contiguous blocks of layers, to test whether grown models form permutable circuits, and how sensitive each method is to late-layer ablations. We then analyse the layer-wise roles within blocks by measuring the similarity between each layer's contribution and the residual stream. Finally, we compare MIDAS with LIDAS on weight symmetry and contribution per attention matrix, connecting it to benchmark results of the previous section. All analyses are conducted on the 1.7B variant described in Table 1 and analogous results for the 360M models are reported in Section E. A detailed description of the setups for each analysis can be found in Appendix Section B. For notation, we follow Csordás et al. [2025]: h_{i+1} denotes the residual stream after transformer layer l_i, a_i the layer's attention output and m_i the output of the MLP.

4.1 Does Depth Growth Lead to Different Depth Utilization?

> **Hypothesis.** Gradual depth-grown Transformers (with MIDAS and LIDAS) utilize model depth more efficiently than conventionally trained, non-grown Transformer baselines.
>
> **Evidence.** Skipping late layers degrades prediction accuracy substantially more for MIDAS and LIDAS than for the baseline, which coincides with an increased depth score.

Experiments. To investigate the contribution of deeper layers, we evaluate intermediate representations via a Tuned Lens [Belrose et al., 2023]. Concretely, for each layer l_i, we train a small affine adapter on a split of FineWeb-Edu [Penedo et al., 2024] that maps that layer's residual output to the hidden representation consumed by the final normalization; we then obtain logits by applying

the model's final normalization and unembedding [Belrose et al., 2023], enabling early-exit at every depth[1]. Subsequently, we quantify depth utilization by the top-5 vocabulary overlap of their predicted vocabularies (Fig. 1B), and early-exiting accuracy on the reasoning primitives (Fig. 1C). Finally, we compute the depth score [Csordás et al., 2025] to summarize where computation occurs along the network by estimating each layer's influence on future tokens (Fig. 1A). For further details, we refer to Appendix Section B, and for results on the 360M model, to Section E.1.

Interpretation. For `MIDAS` and `LIDAS`, Fig. 1B shows that early-exit predictions differ substantially more from the final logits than in the baseline (lower top-5 overlap), indicating that later layers in the grown models add features to the residual stream that are required for the final prediction. In Fig. 1C, the baseline reaches its final performance by Layer 18, whereas accuracy for both grown models continues to improve up to the last layer. Lastly, Fig. 1A reports consistently higher depth scores for the grown models across datasets, most notably on math tasks, showing that more computation is concentrated in later layers.

4.2 Does Depth Growth Form Permutable Computational Blocks?

> **Hypothesis.** Non-grown models depend on their specific layer ordering. Depth-grown models, on the other hand, develop computational blocks that are robust to block-level ordering interventions.
>
> **Evidence.** Reduced performance degradation under multi-layer perturbations indicates lower layer order dependence and greater robustness of `MIDAS` and `LIDAS`.

Experiments. To evaluate layer functional independence, we swap contiguous sub-blocks of sizes $\{1, 2, 4, 8\}$ and measure the effect on downstream performance. While these experiments can indicate robustness, we can also observe how commutative sub-blocks are, as the local order of layers is preserved when swapping larger blocks.

Interpretation. Swapping just single layers does not affect the performance of the baseline and grown models much (Fig. 3), except for the input layers. This observation aligns with findings from Lad et al. [2024]. If we increase the number of consecutive layers that we swap, the accuracy of the baseline quickly starts to deteriorate. In contrast, grown models allow swapping blocks of up to size four with a relatively small decrease in performance, and we observe even less performance degradation when swapping blocks, indicating less order dependence of these blocks. The grown models even reach non-random performance when swapping middle 8-layer blocks compared to the baseline, whose performance drops to random. In general, the degradation is lower on the language-modelling task (Fig. 3 top row) compared to the reasoning primitive (bottom row). Taken together, these effects are most consistent with the emergence of computational blocks whose internal order matters less than the presence of the block as a unit, matching the qualitative behaviour in Fig. 3.

4.3 Does gradually growing form layer-wise patterns?

> **Hypothesis.** The block-wise growing introduces a cyclical pattern in the architecture such that each layer within a block fulfils a certain role.
>
> **Evidence.** The contribution of the attention sublayer, in norm and cosine similarity, repeats in each block. When performing causal interventions, the effect for each layer within a block also repeats. Reversing the order of layers within and especially across blocks destroys the performance of grown models more than swapping, where local order is more preserved.

Experiments. Using the tools of Csordás et al. [2025], we compute for each (sub)layer its cosine similarity to the residual stream ($\frac{a_i \cdot h_i}{||a_i|| ||h_i||}$) and its mean relative contribution $\frac{||a_i||}{||h_i||}$. We then intervene by skipping a transformer layer or sublayer and track the relative changes in downstream computations under two regimes: (i) propagated: zeroing that component's contribution to the residual stream and forwarding this change to all downstream layers; and (ii) local: removing a layer's

[1]Note that this should result in more accurate predictions than naively applying the unembedding matrix at every layer (LogitLens [gebraist, 2020]), as done in Csordás et al. [2025].

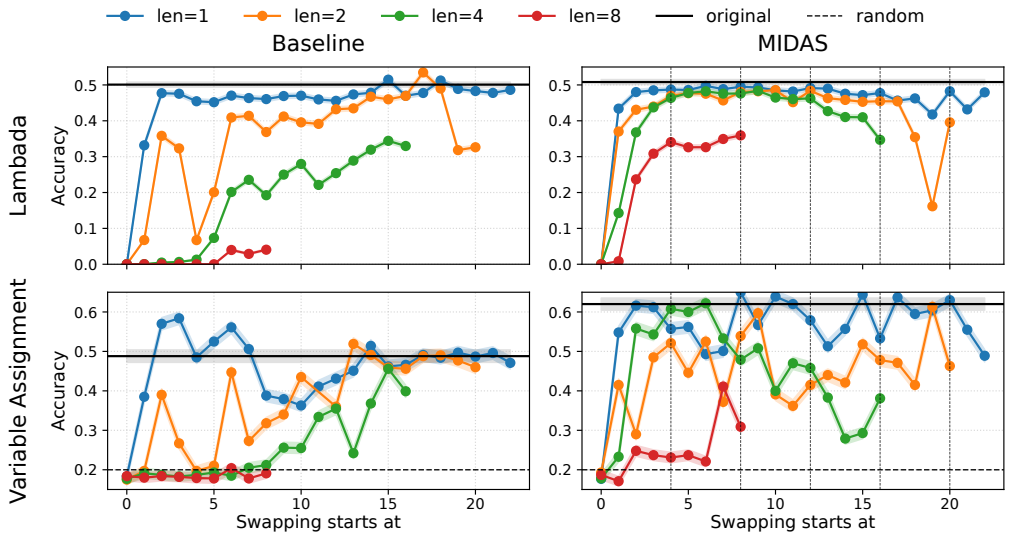


Figure 3: Effect of swapping blocks of layers on Lambada (top row) and the reasoning primitive *Variable Assignment Math* (bottom row). MIDAS is more robust to interventions for larger blocks in the middle of the network: the degradation in performance for MIDAS is much smaller for swapping blocks of larger sizes $\{2, 4, 8\}$ compared to the baseline, especially for Lambada. In Fig. 9 we present results including LIDAS

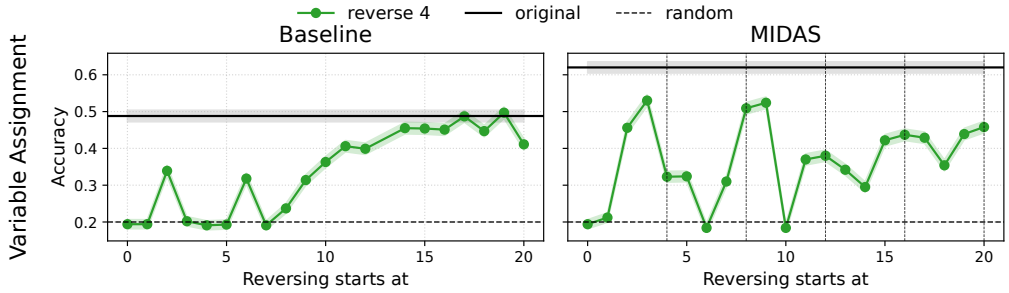


Figure 4: Effect of reversing the order of four consecutive layers on reasoning primitive. Reversing the order of layers within a block (first layer of each grown block as vertical grey lines; right figure) of size 4 degrades the performance for grown models more than swapping the same number of layers (len $= 2$ in Fig. 3). The baseline is more robust to reversing the order of the later layers, while MIDAS is especially sensitive to reversing the order across grown blocks, i.e., the last two and first two layers of consecutive blocks. Starting to reverse at these positions, which correspond to layer index 6, 10, and 14, always results in a drop in performance. Fig. 11 shows results including LIDAS and an additional dataset

contribution from all subsequent inputs separately to isolate pairwise source–target dependencies. Finally, we assess the effect of reversing the order of four consecutive layers and comparing the outcome to results from Fig. 3. A detailed explanation of the interventions can be found in Section B.

Interpretation. Grown models exhibit a highly cyclical pattern in the middle, where the effect is especially visible for the attention sublayer (Fig. 5). The mean relative contribution of the attention sublayer always grows from its lowest point at the first layer of every block to its highest point at the last layer of the block. The highest spike across depth is always at the final layer of the last block in the middle of the network, i.e., the overall second-to-last block. For MIDAS the cosine similarity of the attention sublayers in the middle, similarly to their contributions, always rises from around zero, adding orthogonal features, or slightly negative, weakening or erasing features, to the highest but only slightly positive cosine similarity at the end of each block. The pattern for LIDAS is a little bit less clear, but the cosine similarity never drops as low as MIDAS, potentially adding features from subspaces that are better aligned with the residual stream across the whole block.

Turning towards interventions, by skipping a layer, the most pronounced disruption to future computations arises when skipping the second layer of each block (aside from the earliest layers), with often the biggest observed relative change in the immediate layer after it, i.e., in each block's third layer (Fig. 6a). We hypothesize that the second layer prepares features for future computations. If we measure the relative change on the following layers directly, we notice a clear and striking pattern (Fig. 6b). For future computations, the third layer of every block directly depends on the features of almost all previous layers, potentially performing an aggregating operation. The direct change of removing the output of a previous layer is less on deeper blocks that can depend on more inputs simultaneously, i.e., visually a fading pattern. The last block mostly depends on the final aggregation and strengthening of relevant features performed by the second-to-last block.

Reversing the order of four consecutive layers (Fig. 4) reduces performance in the grown model far more than swapping pairs of two or four layers (len $= 2, 4$ in Fig. 3), where local order is more preserved. The baseline is comparatively robust to reversals involving later layers, which aligns with the hypothesis from Csordás et al. [2025] that later layers in pre-layernorm transformers refine the current output distribution with less order dependence. By contrast, the grown model is most brittle when the reversal straddles block boundaries (last two layers and first two layers of consecutive blocks), showcasing the order of layers within a block matters.

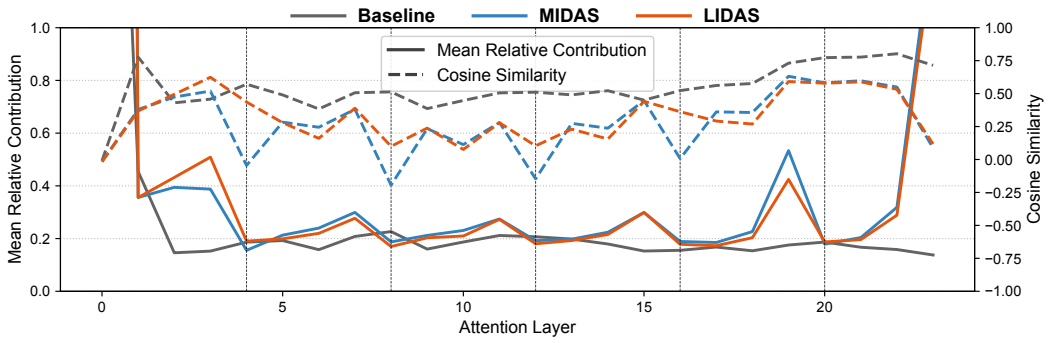
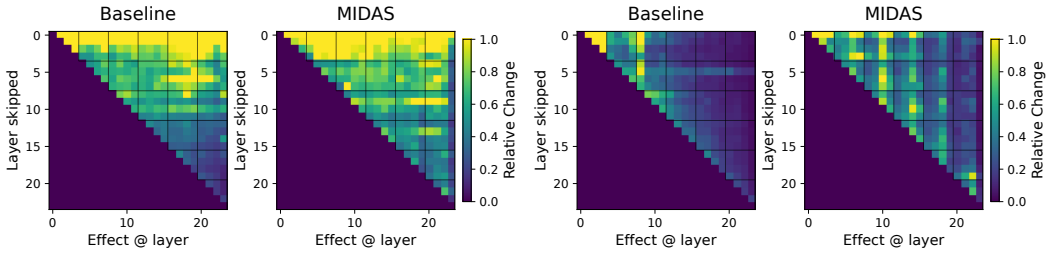


Figure 5: Attention layer contributions to the residual stream. Grown models exhibit a highly cyclical pattern in the centre of the network. The mean relative contribution of the attention sublayer to the residual stream increases throughout a block (whose first layer is denoted by a vertical line) and has its largest contribution in the last layer of each block. While the cosine similarity between the output of each attention layer and the residual stream is relatively flat for the baseline, the pattern for the grown models again depends on the block size and the relative position of the layers within each block. Notably in `MIDAS`, the first attention sublayer of a block has a very low cosine similarity to the residual stream, while for `LIDAS` the attention contributions align more with the residual stream.



(a) Propagated Effect **(b)** Local Effect

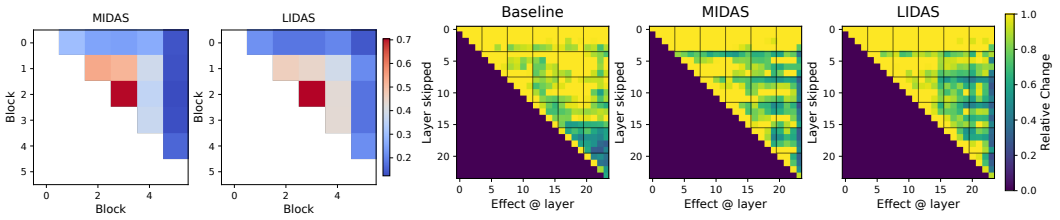
Figure 6: Baseline vs. `MIDAS`. Effect of skipping a layer on downstream layer contributions for *future* tokens. (a) `MIDAS` relies more on later layers than the baseline for future computations. Especially skipping the second layer of each mid-block strongly impacts the immediately following layer. (b) For `MIDAS`, the third layer of every block in the middle directly depends on all previous computations. We refer to Fig. 12 and Fig. 13 for results including `LIDAS`.

> **Hypothesis.** Compared to MIDAS, LIDAS produces more symmetric weights and utilises its depth more effectively, making better use of central layers towards better empirical performance.
>
> **Evidence.** In LIDAS, inter-block cosine similarities are higher and more symmetric about the centre. Skipping the first attention sublayer in the middle blocks causes larger relative changes in the hidden state of the token under consideration.

Experiments. To measure the weight similarity of blocks for the grown model, we concatenate all weights from the feedforward layers of a block and calculate the cosine similarity to other blocks. Similarly to before, we skip layers and measure the relative change for all later layers, but now on *all* tokens (including the current token).

Interpretation. In LIDAS we observe a block-similarity structure that is symmetric about the model's centre, whereas in MIDAS the central block is more similar to the earlier (upper) blocks than to the later (lower) ones, yielding an asymmetric pattern (Fig. 7). This difference follows from the growth rule: LIDAS duplicates the exact layer-wise middle, while MIDAS is constrained to the nearest block centre. With an even number of blocks, the MIDAS choice necessarily biases similarity toward one side.

Additionally, this growing strategy leads to a higher utilization of the first attention sublayer of every block (Fig. 7b), making it more aligned with the residual stream and having a bigger effect on the current computations of future layers.



(a) Block similarity **(b)** Effect of skipping attention sublayer

Figure 7: Baseline vs. MIDAS vs. LIDAS. (a) The weight similarity, measured by cosine similarity, between feedforward layers per block is more symmetric for LIDAS compared to MIDAS. We omit the baseline as its weight similarities are all close to zero. (b) Skipping the first attention sublayer of every block in the centre of the network has a lower effect on the following layers' current computations in MIDAS compared to LIDAS.

5 Discussion

This work systematically investigates how gradual depth growth in large language models affects their computational dynamics, providing a mechanistic explanation for their improved reasoning performance. Our findings confirm that gradually grown models, outperform conventionally trained baselines on reasoning tasks and in training cost. Through detailed analysis, we demonstrate that this performance is tied to a more effective utilization of model depth. Unlike non-grown models that suffer from a Curse of Depth [Sun et al., 2025b; Csordás et al., 2025], our grown models continue to perform novel computations in their later layers and exhibit a higher overall depth score. We show that this is enabled by the formation of permutable computational blocks in the middle of the network, with each layer within these blocks serving a distinct cyclical role. The superiority of our proposed lightweight and novel stacking variant LIDAS is attributed to its ability to create a more symmetric weight structure and more effective attention layers, leading to improved robustness to interventions. In conclusion, our research provides critical insights into the internal workings of depth-grown models, confirming that these training procedures can overcome key architectural inefficiencies and pave the way for more efficient and capable model development.

REPRODUCIBILITY STATEMENT

We conduct all training and experiments using publicly available code and datasets. Our training setup builds on the open-source `nanotron`[2] library; Table 2 lists all model and optimizer hyperparameters, and Section A provides the exact SmolLM training mixture with links to each public dataset to fully reconstruct the training corpus. Additionally, we provide a detailed description of the growing operators in Section 3.1 and further detail in Section A. To reproduce our analyses, Section B details the evaluation protocols and the open-source libraries we use, along with any task-specific settings.

REFERENCES

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, et al. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025.

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

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. SmolLM-corpus, July 2024. URL `https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus`.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 2006.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. *arXiv preprint arXiv:2502.13842*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber, Christopher Potts, and Christopher D Manning. MOEUT: Mixture-of-experts universal transformers. *Advances in Neural Information Processing Systems*, 37:28589–28614, 2024.

Róbert Csordás, Christopher D Manning, and Christopher Potts. Do language models use their depth efficiently? *arXiv preprint arXiv:2505.13898*, 2025.

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

Wenyu Du, Tongxu Luo, Zihan Qiu, Zeyu Huang, Yikang Shen, Reynold Cheng, Yike Guo, and Jie Fu. Stacking your transformers: A closer look at model growth for efficient LLM pre-training. *arXiv preprint arXiv:2405.15319*, 2024.

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`.

[2] `https://github.com/huggingface/nanotron`

gebraist, 2020. URL lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Jonas Geiping, Sean 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. *arXiv preprint arXiv:2502.05171*, 2025.

Andrea Gesmundo and Kaitlin Maile. Composable function-preserving expansions for transformer architectures. *arXiv preprint arXiv:2308.06103*, 2023.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023.

Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu. Efficient training of BERT by progressively stacking. In *International conference on machine learning*, pp. 2337–2346. PMLR, 2019.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset, 2021.

Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*, 2023.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, pp. 1152–1157, 2016.

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

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2019.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*, 2024.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*, 2021.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. OpenWebMath: An open dataset of high-quality mathematical web text. *arXiv preprint arXiv:2310.06786*, 2023.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

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

Jackson Petty, Sjoerd van Steenkiste, Ishita Dasgupta, Fei Sha, Dan Garrette, and Tal Linzen. The impact of depth and width on transformer language model generalization. *CoRR*, 2023.

Sashank J Reddi, Sobhan Miryoosefi, Stefani Karp, Shankar Krishnan, Satyen Kale, Seungyeon Kim, and Sanjiv Kumar. Efficient training of language models using few-shot learning. In *International Conference on Machine Learning*, pp. 14553–14568. PMLR, 2023.

Nikunj Saunshi, Stefani Karp, Shankar Krishnan, Sobhan Miryoosefi, Sashank Jakkam Reddi, and Sanjiv Kumar. On the inductive bias of stacking towards improving reasoning. *Advances in Neural Information Processing Systems*, 37:71437–71464, 2024.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. Transformer layers as painters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25219–25227, 2025a.

Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. *arXiv preprint arXiv:2502.05795*, 2025b.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pp. 2464–2469. IEEE, 2016.

Gaël Varoquaux, Sasha Luccioni, and Meredith Whittaker. Hype, sustainability, and the price of the bigger-is-better paradigm in AI. In *Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency*, pp. 61–75, 2025.

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

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie, Charlotte Frenkel, Razvan Pascanu, Blaise Agüera y Arcas, and João Sacramento. Mesanet: Sequence modeling by locally optimal test-time training, 2025. URL https://arxiv.org/abs/2506.05233.

Kevin Wang, Ishaan Javali, MichaŁ Bortkiewicz, Benjamin Eysenbach, et al. 1000 layer networks for self-supervised RL: Scaling depth can enable new goal-reaching capabilities. *arXiv preprint arXiv:2503.14858*, 2025.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. *arXiv preprint arXiv:2303.00980*, 2023.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*, 2020.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. *arXiv preprint arXiv:2311.12424*, 2023.

Yiqun Yao, Zheng Zhang, Jing Li, and Yequan Wang. Masked structural growth for 2x faster language model pre-training. *arXiv preprint arXiv:2305.02869*, 2023.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, et al. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*, 2023.

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

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions, 2023.

DISCLOSURE OF LLM USAGE

Large language models were used for language editing, such as enhancing clarity, precision, and flow, and for minor aesthetic adjustments to figures to improve interpretability.

A SMOLLM: ARCHITECTURE & DATA

Data. For all SmolLM models we trained, we followed the SmolLM-v1 data mixture from Ben Allal et al. [2024].

- **FineWeb-Edu (dedup)** [Ben Allal et al., 2024]: Educational slice of FineWeb selected with a Llama3-70B–trained "educational quality" classifier. We use the deduplicated subset (≈220B tokens) included in the SmolLM-Corpus.
- **Open-Web_Math** [Paster et al., 2023]: High-quality mathematical web pages extracted from Common Crawl with math-aware parsing, quality filtering, and deduplication (≈14.7B tokens). Used to enrich math/reasoning coverage.
- **Cosmopedia-v2** [Ben Allal et al., 2024]: Synthetic textbooks, stories, and code generated with Mixtral-8×7B using curated topic lists and seed pages. v2 totals ≈39M documents (≈28B tokens of textbooks/stories).
- **Python-Edu** [Ben Allal et al., 2024]: Educational Python subset built by training an "educational code" classifier on annotated samples from *The Stack* and applying it to the StarCoder training corpus. It contains of ≈4B tokens with strict quality thresholding.

Given a fixed training-token budget, we then sample the corpus by proportion—70% FineWeb-Edu (deduplicated), 15% Cosmopedia-v2, 9% Python-Edu, 6% OpenWebMath. Note that this leads to significant upsampling of the smaller datasets like Python-Edu and OpenWebMath.

Model architecture. Both sizes follow a LLaMA-style, decoder-only Transformer with RMSNorm, SwiGLU MLPs, and RoPE positional embeddings (tied input/output embeddings). The 360M variant uses GQA.

	SmolLM-1.7B	**SmolLM-360M**
Layers	24	32
Model width	2048	960
FFN dimension	8192	2560
Attention heads	32	15
KV heads	32 (MHA)	5 (GQA)
Norm	RMSNorm	RMSNorm
MLP activation	SwiGLU	SwiGLU
Batch size	2M	1M
Learning rate $\eta_{\max}$	0.0005	0.003
Weight decay	0.01	0.01
Positional embeddings	RoPE (θ=10,000)	RoPE (θ=100,000)
Context length (pretrain)	2048	2048
Tokenizer	cosmo2 [3]	cosmo2
Tied embeddings	Yes	Yes

Table 2: Hyperparameters for both SmolLM models

Training. We train both sizes for 200k iterations. This corresponds to roughly 200B seen tokens for the 360M model and 400B for the 1.7B model. We use a trapezoidal learning-rate schedule with a linear warmup for the first 2000 steps up to the peak rate $\eta_{\max}$, a constant plateau until step 170000, and a square root decay over the final 30000 steps. We optimise with AdamW [Loshchilov & Hutter, 2019] and apply global gradient clipping at 1.0 for all runs.

Training with the Growing operator For SmolLM with gradual depth growth all training hyperparameters match the baseline in Table 2. We use a fixed block size $b = 4$ and insert a new *middle* block after each stage, instantiating either MIDAS (duplicate the middle stage block) or LIDAS (duplicate the layer-wise middle; see Section 3.1), while keeping width and attention heads constant. At every growth step we deep-copy all layer parameters and their optimizer state so duplicated layers start identically (same AdamW moments) and then diverge with continued training; embeddings and the final head are copied unchanged. The number of growth stages is defined by $k = L_{\text{final}}/b$ and let T be the total training steps. We allocate per-stage budgets using the PROP-α schedule of Saunshi et al. [2024]:

$$T_i \;=\; \frac{i^{\alpha}}{\sum_{j=1}^{k} j^{\alpha}}\, T \quad \text{for } i = 1, \ldots, k,$$

and use PROP-1 (α=1) in our experiments. In practice, we round T_i to integers (largest-remainder to keep $\sum_i T_i = T$) and maintain a *single continuous* learning-rate schedule across stages (no LR reset; the scheduler's global step carries over). We set $T = 170{,}000$ so all models reach their final depth before they enter the cooldown phase.

Compute requirements. We trained all models on NVIDIA A100 GPUs (40 gb). The large (static baseline) model ran on 128 GPUs for 4.5 days, and the small (static baseline) model ran on 64 GPUs for 1.5 days.

B Experimental Setup

Codebases and provenance. Depth analyses and interventions follow the methodology of Csordás et al. [2025], extended to block-wise skip/swap over consecutive layers (block sizes $\{1, 2, 4, 8\}$) and further extended to permuting consecutive layers in arbitrary order. Tuned Lens experiments follow Belrose et al. [2023]. For reproducibility, we adhere to the default configurations and the fixed GSM8K prompt subset used in the original depth-analysis setup.

Reproducibility defaults. We adopt the default configuration from the original depth-analysis repository of Csordás et al. [2025] for reproducibility. Specifically, we use the same fixed set of GSM8K prompts/examples for early-exit, skip, swap and relative contribution evaluations, and we keep random seeds, batching, and evaluation hyperparameters at their defaults unless stated otherwise.

Models and data. We analyze SmolLM-v1 backbones at 360M and 1.7B parameters (training details in Section A) and evaluate on MATH, MQuAKE, and GSM8K as described in the main text. Preprocessing follows Csordás et al. [2025].

Intervention protocols. We distinguish *heatmap (relative-change) experiments* from *benchmarked interventions*. Heatmaps quantify relative changes and use **single (sub)layer skipping only**. Benchmarked interventions (accuracy-based) are described separately below. For heatmaps, we evaluate two intervention *modes* and two *measurement axes*, following and extending Csordás et al. [2025]:

- **Current vs. future effects.** In the *current* setting, we intervene by erasing the entire (sub)layer contribution *for all tokens* and measure changes on all positions. In the *future* setting, for a chosen boundary token index t, we erase the (sub)layer contribution *only for tokens* $\leq t$, leaving tokens $> t$ unchanged at that (sub)layer; we then measure changes *strictly on tokens* $> t$. This design directly tests whether information is transferred to later tokens via attention, ruling out purely pointwise (self-only) computation.
- **Output vs. later-layer effects.** For the *output probability distribution*, we compute the L2 norm difference between the softmaxed logits of the intervened and original forward passes, aggregated over the relevant positions (current or future). For the *later-layer effects*, we compute, for each later layer, the *relative change* in the residual contribution (i.e., the norm of the difference in that layer's residual update divided by the norm of the original residual update), again aggregated over the relevant positions.

Concretely for heatmaps, in the *future* effects evaluation we select multiple boundary indices t and, for each t, (i) erase the (sub)layer's contribution only at tokens $\leq t$, (ii) keep its contribution intact at tokens $> t$, and then compare the intervened and original runs on (a) softmaxed output distributions at positions $> t$ and (b) residual contributions of all later layers at positions $> t$. This directly tests whether features are moved forward in time (to future tokens) by attention.

For heatmaps, we also include a **local (direct) effects** variant, which isolates pairwise dependencies between a source layer and a later target layer without allowing effects to *propagate* through multiple subsequent layers. Specifically, for a source layer s and a later layer $\ell > s$, we subtract the stored contribution of s from the residual fed into ℓ and record the relative change at ℓ; we do *not* roll this modification forward beyond ℓ. This complements the propagated analyses by revealing direct, non-compounded influences.

Heatmap interventions are performed at the layer or sublayer level and are **strictly single-layer**. The current/future distinction applies *only* to these heatmap experiments. Block-wise operations are used solely in benchmarked interventions (below).

Aggregation for heatmaps. For heatmap visualizations of later-layer effects, we aggregate by taking the *maximum* relative change across (i) batch examples, (ii) eligible sequence positions, and (iii) multiple chosen boundaries t in the future setting. Concretely, for current effects we take the max over all positions; for future effects we take the max only over positions strictly greater than t, and then take the max over all tested t for each example. This yields a single matrix of source-layer by target-layer maxima per model/setting.

Tuned Lens training and evaluation. Following Belrose et al. [2023], we train, for each layer, a small affine adapter that maps that layer's residual output to the hidden representation with the same shape that serves as the input to the *final* normalization layer immediately before the unembedding. Final logits are then obtained by applying the model's final normalization and unembedding as usual. Adapters are trained on a held-out split of FineWeb-Edu and evaluated by (a) KL divergence between early-exit and final distributions and (b) top-5 vocabulary overlap with the final prediction (cf. Fig. 1B for 1.7B and Fig. 15 360M).

Benchmarked interventions. We evaluate accuracy on downstream benchmarks under: (i) Tuned Lens early-exit (using the adapter path described above), (ii) skip interventions, and (iii) swap interventions. For benchmarks, we may intervene on contiguous **blocks** of sizes $\{2, 4, 8\}$ (in addition to single layers). We decode with greedy top-1 and compute benchmark accuracy (e.g., MathWorld, reasoning primitives), matching the evaluation protocol used for the unmodified model. The current/future distinction does *not* apply to benchmark evaluations.

Depth score. We report the *logit-effect* depth score based on `mean_dout`. For each layer ℓ, `mean_dout` is the across-examples mean of the maximum L2 change in the softmaxed output distribution at future tokens when intervening at layer ℓ (future-setting; see intervention protocols). We normalize this per-layer vector to a probability distribution over layers and take its expected layer index as the depth score.

C Detailed Benchmark Results

In Section 3.2 we have shown aggregated results over several Benchmarks. In this section, we present the Detailed results for all models. We have evaluated out models on these benchmarks using the language model evaluation harness library [Gao et al., 2024]. The Reasoning primitives we have implemented ourselves following Saunshi et al. [2024]. Induction copying is generated by sampling a sequence of random 3-letter words (e.g., length 10), selecting a contiguous subsequence (e.g., length 5) from within it, appending that subsequence, and asking for the next token in the original sequence. Variable assignment is generated by sampling variable–value statements and querying a single variable's value. We use the authors' basic/math/code prompt templates.

In Tables 3 and 4 we report per-dataset results for Open-Book and Closed-Book QA. In line with Saunshi et al. [2024], both grown models (`MIDAS` and `LIDAS`) yield larger gains on Open-Book QA than on Closed-Book QA. Notably, `LIDAS` 1.7B improves over the 1.7B baseline even on most

		CoQA	DROP	QuAC	SquadV2	TyDi QA (wc)
360M	Baseline	46.08	12.48	14.27	24.35	17.25
	Midas	50.00	12.75	14.10	24.96	21.06
	Lidas	**51.50**	**15.25**	**15.79**	**28.11**	**22.50**
1.7B	Baseline	58.36	16.52	15.91	33.88	**23.17**
	Midas	59.35	16.88	17.30	36.06	14.39
	Lidas	**63.41**	**17.66**	**17.91**	**36.56**	13.65

Table 3: Open-book QA Benchmarks.

Closed-Book datasets and remains competitive on the rest, which differs from observations made with MIDAS

		Trivia QA	Web Questions	TyDi QA (nc)	Natural Questions
360M	Baseline	19.23	**16.78**	**12.98**	9.01
	Midas	18.90	14.58	12.64	8.89
	Lidas	**20.80**	15.61	12.14	**9.73**
1.7B	Baseline	27.72	19.20	15.34	12.18
	Midas	**27.98**	17.96	16.16	11.91
	Lidas	26.85	**20.24**	**16.34**	**12.90**

Table 4: Closed-book QA Benchmarks.

		ASDiv	**MAWPS** Add/Sub	**MAWPS** Multi-Arith	**MAWPS** Single-Op	**MAWPS** Single-Eq	**SVAMP**
360M	Baseline	3.34	**3.67**	**1.72**	5.66	2.75	**5.02**
	Midas	3.77	**3.67**	1.15	6.29	**6.42**	**5.02**
	Lidas	**4.64**	1.83	**1.72**	**7.55**	**6.42**	4.01
1.7B	Baseline	11.15	14.68	1.15	25.16	**22.02**	8.36
	Midas	12.93	18.35	2.30	33.96	20.18	8.70
	Lidas	**14.88**	**25.69**	**2.87**	**38.36**	18.35	**11.37**

Table 5: Math World.

On the reasoning benchmarks, Math World (Table 5) and Reasoning Primitives (Table 6), improvements at 360M are modest on average, while at 1.7B they become more pronounced. For Math World, LIDAS 1.7B attains the best score on five of six subsets (the exception is MAWPS Single-Equation). For Reasoning Primitives, both MIDAS and LIDAS surpass the baseline, with LIDAS 1.7B leading on copying and on the code/math variable-assignment formats, while MIDAS slightly edges LIDAS on the basic variable-assignment format.

In addition to improved reasoning performance, models trained with gradual stacking also reduce the computational resources needed. Specifically, MIDAS and LIDAS only require $\sim 77\%$ of the FLOPs used to train the baseline Table 7.

D Supplementary: Big Models (1.7B)

D.1 Benchmark ablations

We report accuracy under four benchmarked interventions on *Lambada* and the *Variable Assignment d0 MC* primitive: (i) early exit via Tuned Lens (adapter-to-final-readout), (ii) swapping contiguous

		Copying (random words)	**Copying** (real words)	**Variable assignment** (basic)	**Variable assignment** (code)	**Variable assignment** (math)
360M	Baseline	**15.50**	13.30	20.50	**58.30**	42.70
	Midas	13.80	14.10	20.00	52.90	40.10
	Lidas	14.20	**19.70**	**24.30**	51.80	**46.00**
1.7B	Baseline	16.80	23.60	20.80	64.20	48.80
	Midas	19.30	24.60	**37.00**	61.50	62.00
	Lidas	**28.40**	**31.00**	36.70	**71.80**	**68.80**

Table 6: Reasoning Primitives.

Model	PetaFLOPs	Ratio
360M Standard	613527.488	1.289
360M Grown	476147.897	1.000
1700M Standard	4813222.102	1.288
1700M Grown	3736608.182	1.000

Table 7: PetaFLOPs used for training 200k iterations

blocks, (iii) skipping single layers or blocks, and (iv) reversing the order of four consecutive layers (see Figs. 8 to 11).

D.2 Relative contribution heatmaps

We visualize relative changes under single-(sub)layer skipping in two future-effect variants—propagated and local—and a current-effect attention ablation (see Figs. 12 to 14).

E Supplementary: Small Models (360M)

E.1 Depth summary

We summarize small-model depth utilization using the depth score and tuned-lens early-exit diagnostics; see Fig. 15.

E.2 Benchmark ablations

We replicate benchmarked interventions at 360M to assess robustness under reduced capacity, covering early exit, swap, skip, and reversal (see Figs. 16 to 19).

E.3 Relative contribution heatmaps

We include the small-model counterparts of the future propagated, future local, and current attention ablations (see Figs. 20 to 22).

E.4 Additional plots

For completeness, we show block-similarity structure at 360M, which mirrors the symmetry patterns observed at 1.7B (cf. Fig. 7a).

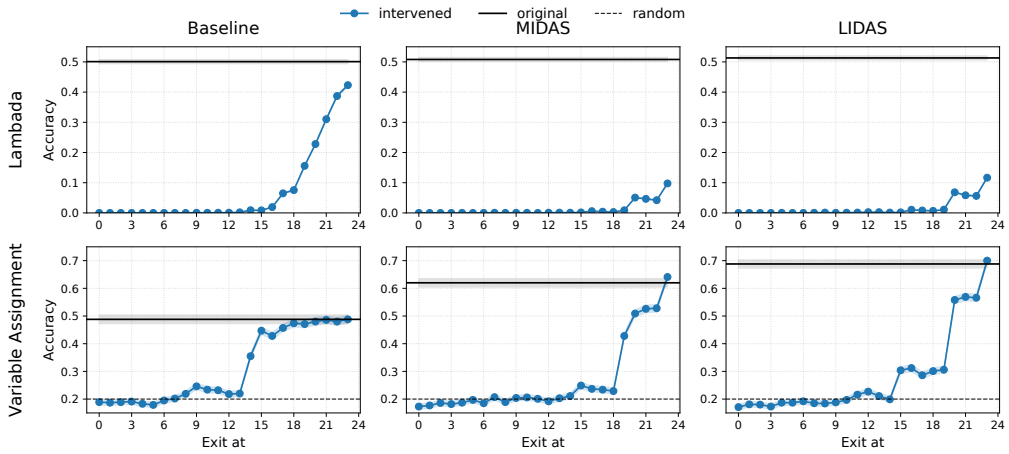


Figure 8: Big models (1.7B): early exit with tuned lens on *Lambada* and *Variable Assignment d0 MC* for Baseline, MIDAS, and LIDAS.

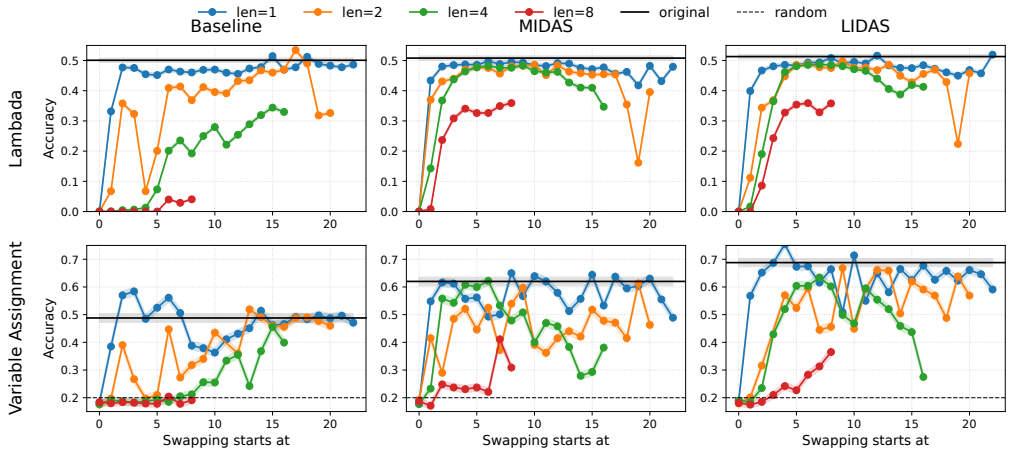


Figure 9: Big models (1.7B): swap ablations on *Lambada* and *Variable Assignment d0 MC* for Baseline, MIDAS, and LIDAS.

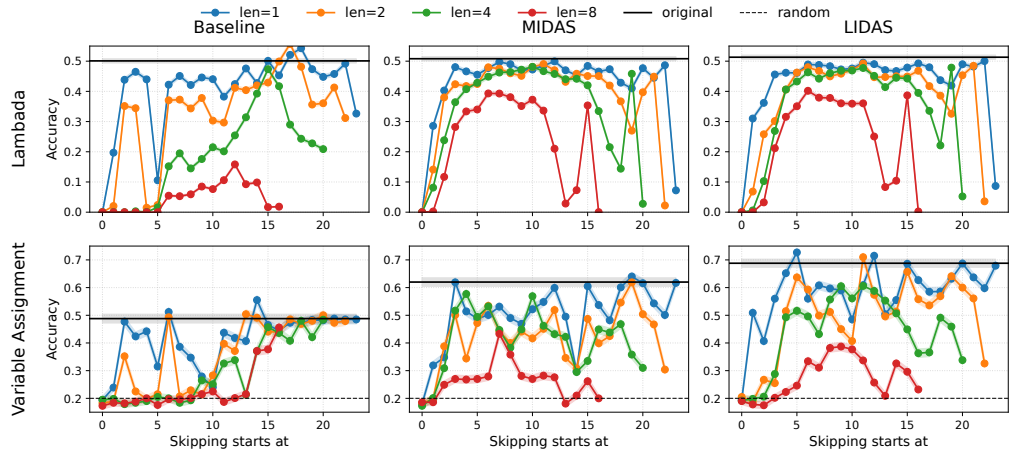


Figure 10: Big models (1.7B): skip ablations on *Lambada* and *Variable Assignment d0 MC* for Baseline, MIDAS, and LIDAS.

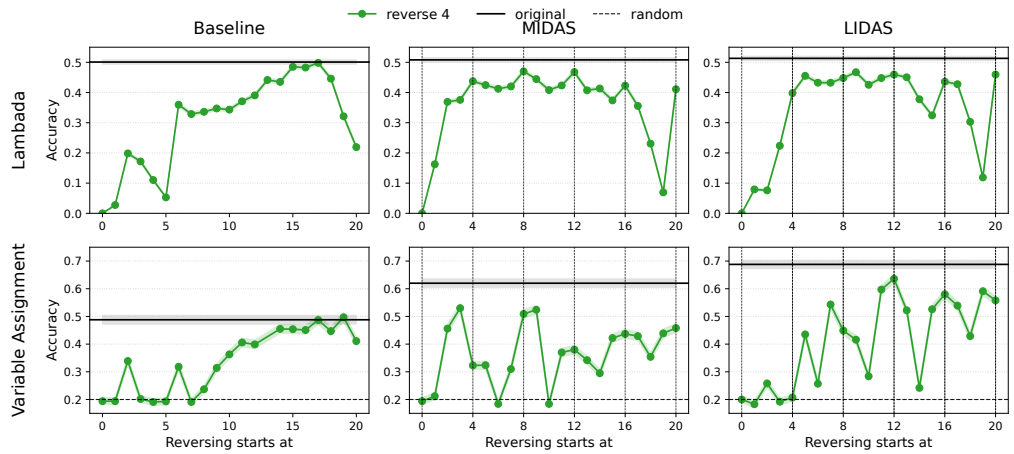


Figure 11: Big models (1.7B): reversing the order of 4 consecutive layers on *Lambada* and *Variable Assignment d0 MC* for Baseline, MIDAS, and LIDAS.

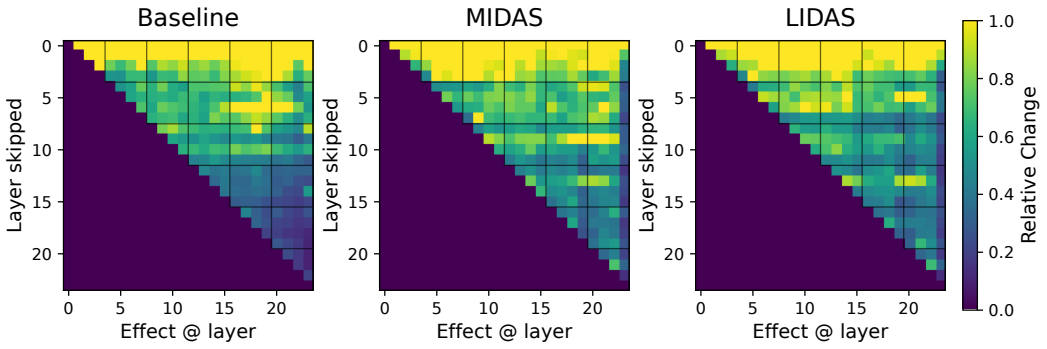


Figure 12: Big models (1.7B): propagated future effects of single-layer skipping.

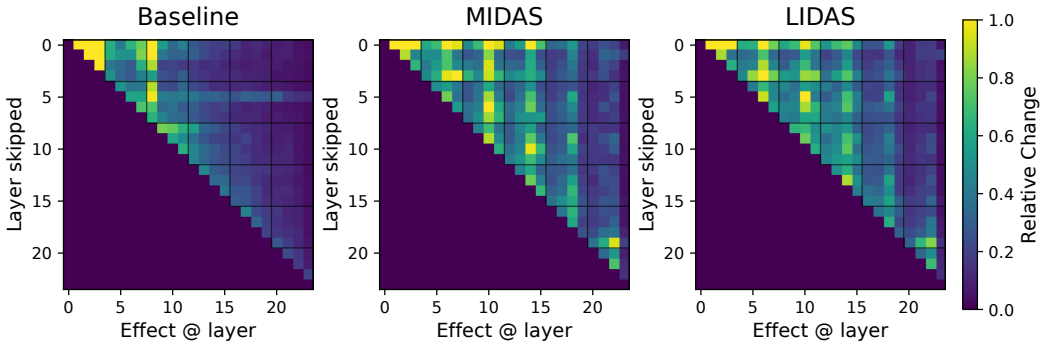


Figure 13: Big models (1.7B): local future effects of single-layer skipping.

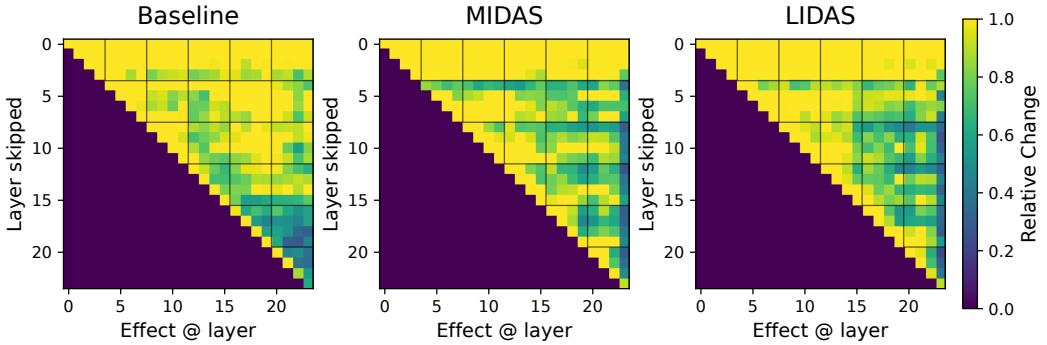


Figure 14: Big models (1.7B): current effects when skipping the attention sublayer.

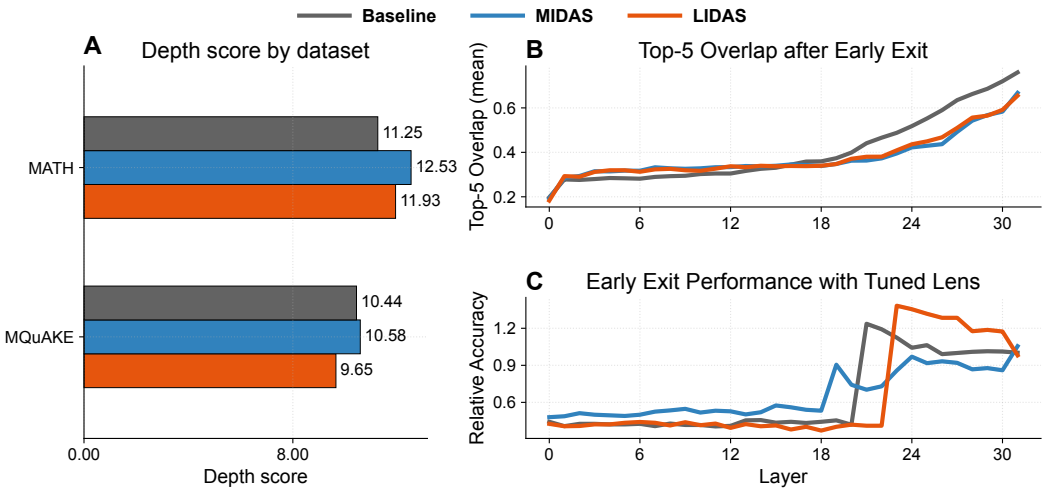


Figure 15: Depth-grown models use their depth more. Results are less pronounced for smaller models.

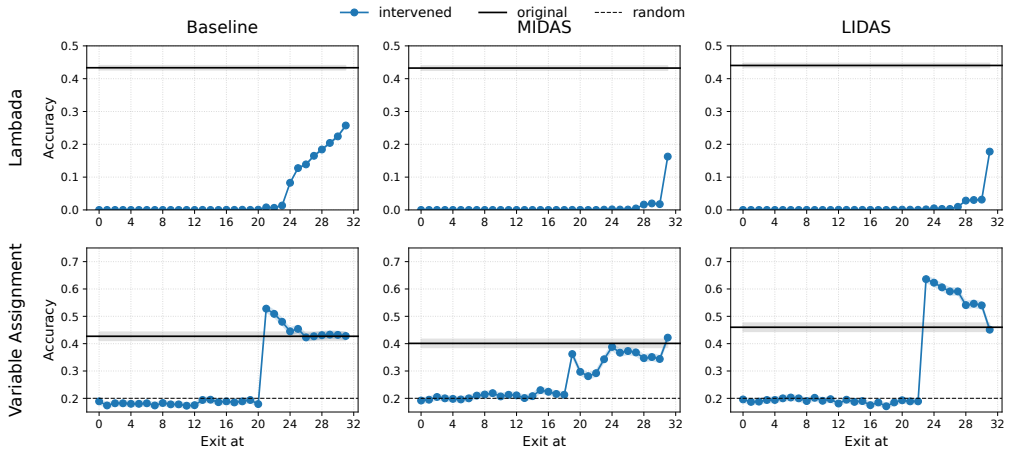


Figure 16: Small models (360M): early exit with tuned lens on *Lambada* and *Variable Assignment d0 MC* for Baseline, `MIDAS`, and `LIDAS`.

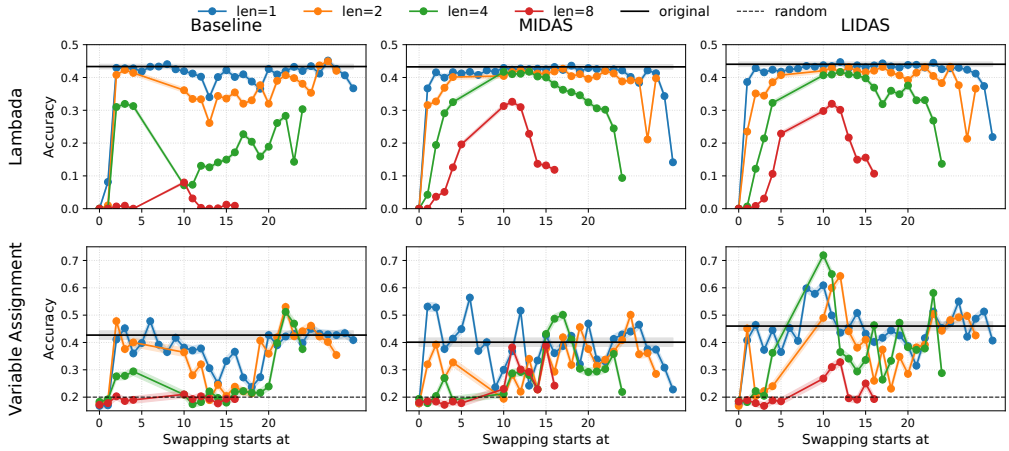


Figure 17: Small models (360M): swap ablations on *Lambada* and *Variable Assignment d0 MC*.

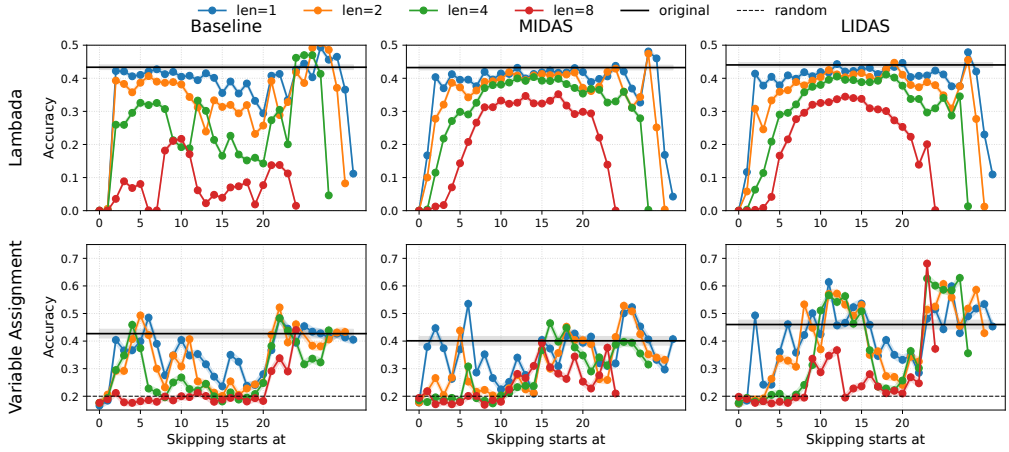


Figure 18: Small models (360M): skip ablations on *Lambada* and *Variable Assignment d0 MC*.

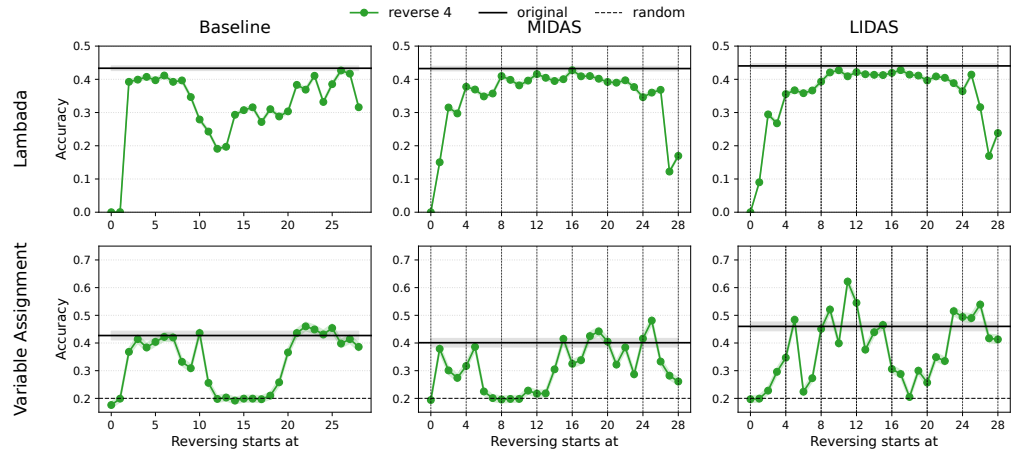


Figure 19: Small models (360M): reversing the order of 4 consecutive layers on *Lambada* and *Variable Assignment d0 MC*.

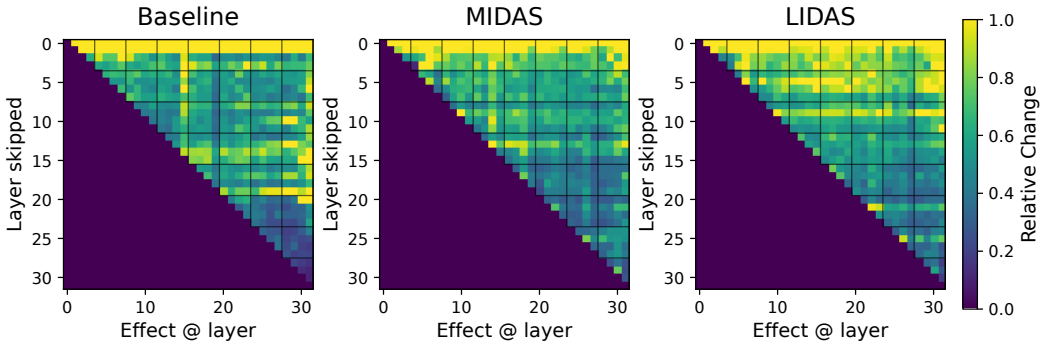


Figure 20: Small models (360M): propagated future effects of single-layer skipping.

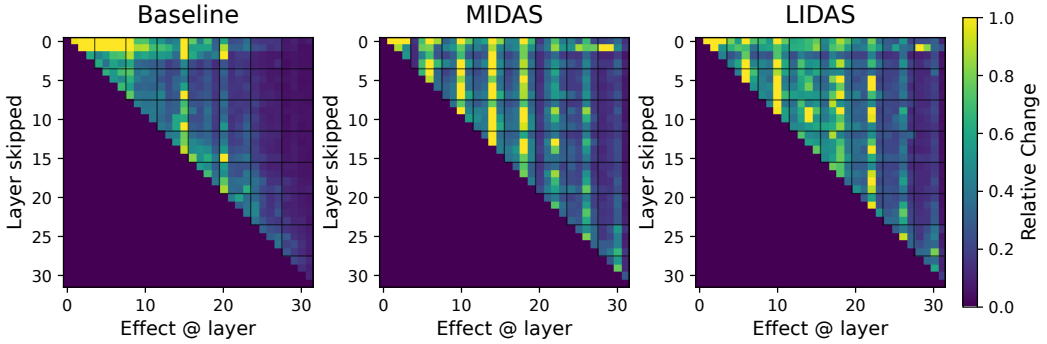


Figure 21: Small models (360M): local future effects of single-layer skipping.

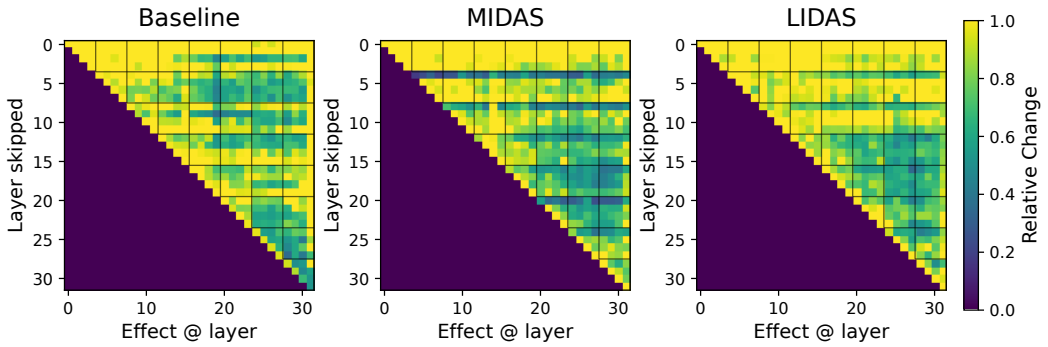


Figure 22: Small models (360M): current effects when skipping the attention sublayer.

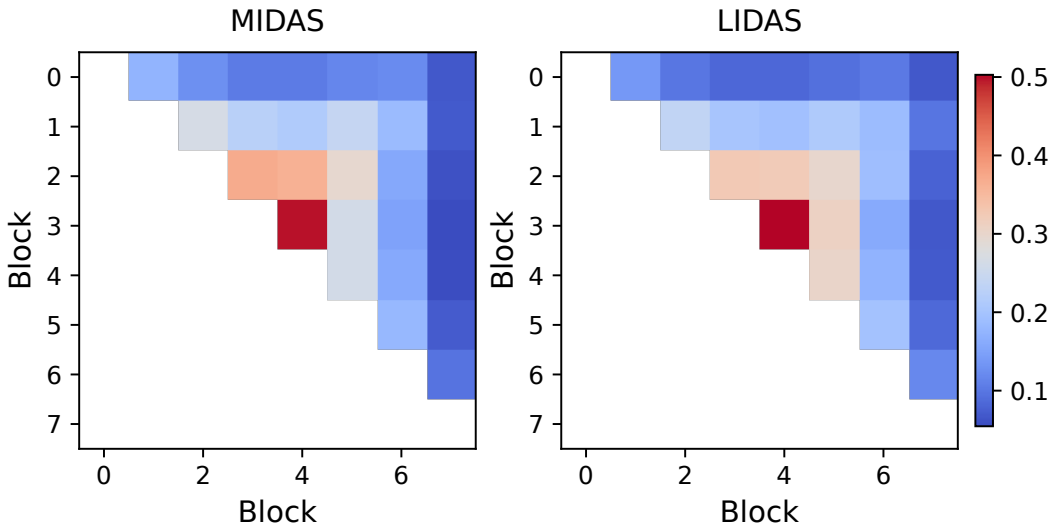


Figure 23: Small models (360M): block similarity for `MIDAS` and `LIDAS`.