Pretraining Builds Cooperation, Not Components: Tracking Inter-Layer Coordination via Lie Brackets

JaeSeong Kim
Semyung University
Jecheon-si
Republic of Korea
kjsqp1010@semyung.ac.kr

Suan Lee
Semyung University
Jecheon-si
Republic of Korea
suanlee@semyung.ac.kr

Abstract

Transformer language models predict the next token by passing through dozens of sequential layers, yet whether adjacent layers operate independently or coordinate their computations—and how this relationship forms during pretraining—remains largely unknown. We view residual updates of adjacent layers as vector fields and introduce a Lie bracket-based **inter-layer coordination** metric that quantifies how much two layers depend on each other's outputs. Analyzing pretraining checkpoints of 6 models across 3 architecture families (Pythia, OLMo, Amber), we find that all layer pairs are nearly independent at initialization, but as pretraining progresses, individual layer contributions decrease while coordination systematically increases—concentrating in the middle layers that existing metrics (BI, CKA) classify as "redundant." The mechanism is Jacobian amplification: middle layers write small updates but respond sensitively to their neighbors, and this amplification co-emerges with coordination ($r > 0.99$). Block duplication experiments confirm that high-coordination blocks are 2.7–7.9× more robust than low-coordination blocks, verifying that this cooperative structure substantively contributes to model function. Our findings reframe pretraining as a process that builds cooperative circuits between layers rather than strengthening each layer individually.

1 Introduction

Recent studies have analyzed the roles played by individual layers in transformer language models (Belinkov, 2022; Kornblith et al., 2019). Probing analyzes the linguistic information encoded at each layer, Block Influence measures the update magnitude of each layer (Men et al., 2024), and CKA quantifies representational similarity between layers (Kornblith et al., 2019). Meanwhile, circuit-level analyses have identified inter-layer pathways responsible for specific behaviors (Elhage et al., 2021; Olsson et al., 2022). However, little is known about how much adjacent layers depend on each other's outputs and how this dependency forms during pretraining—that is, the emergence process of inter-layer coordination structures. In this work, we analyze how layers transition from independent computation to cooperative computation over the course of pretraining, and what this transition implies for the model's internal organization. We hypothesize that pretraining does not strengthen each layer individually, but rather builds cooperative structures between layers.

We introduce **inter-layer coordination**, which reflects how much adjacent layers depend on each other's outputs, and investigate how this relationship changes during pretraining. Each layer in a transformer follows a residual connection structure $f_i(x) = x + g_i(x)$ (He et al., 2016), and there are limits to the computation each layer can perform on its own. Motivated by this observation, we view the residual updates of two adjacent layers as vector fields and quantify the output change when their execution order is swapped, based on the Lie bracket. When the Lie bracket is nonzero, there exists a new computational direction that emerges only when the two layers interact. By normalizing this quantity by each layer's individual update magnitude, we isolate the pure degree of coordination, yielding the inter-layer coordination metric. Low coordination indicates that the two layers operate independently, while high coordination indicates that the layers have evolved to perform computations through cooperation that are individually unattainable. We analyze pretraining checkpoints of 6 models across 3 architecture families (Pythia, OLMo, Amber)

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 2952–2964
November 6-10, 2026 ©2026 Association for Computational Linguistics

to track how coordination changes throughout pretraining. Our analysis reveals that immediately after initialization, coordination is uniform across all layer pairs, but as pretraining progresses, the coordination of middle layers increases systematically.

Analyzing how this increase in coordination relates to patterns captured by existing metrics, we discover that the regions where coordination forms most strongly coincide precisely with the regions that existing metrics classify as "redundant." In the middle layers where Block Influence is minimal and CKA approaches 1.0—regions judged to "do almost nothing" individually—inter-layer coordination is at its maximum. This inversion does not exist at initialization and emerges through pretraining, suggesting that pretraining changes the model not by increasing each layer's individual contribution but by building inter-layer cooperation.

We hypothesize that the mechanism underlying this coordination structure is Jacobian amplification. Middle layers have extremely small individual outputs, yet their input sensitivity reaches up to 53 times their output magnitude; swapping the order delivers a different input to these high-sensitivity layers, producing disproportionately large changes. To verify this hypothesis, we track the learning trajectory of amplification ratios and confirm that amplification and coordination emerge at the same point in training ($r > 0.99$). Furthermore, to validate whether this coordination structure substantively contributes to model function, we conduct a block duplication sweep experiment. When blocks with high coordination—blocks where inter-layer cooperation is strong—are duplicated, the model remains more robust, a finding confirmed by significant correlations across the majority of models. To the best of our knowledge, this is the first study to define pairwise inter-layer coordination in pretrained language models and systematically track how this coordination structure emerges over the course of pretraining.

2 Inter-Layer Coordination

To understand how layers process computation at each stage of pretraining, we introduce a new concept called **inter-layer coordination**. Inter-layer coordination quantifies how much two adjacent layers depend on each other's outputs—that is, how much they "attend to" one another.

2.1 Independent Layers and Cooperative Layers

Each layer in a transformer follows a residual connection structure $f_i(x) = x + g_i(x)$, where $g_i(x)$ is the update that layer i applies to input x. If two layers are completely independent—each performing its own computation regardless of its input—then swapping their execution order leaves the final output unchanged: $h + g_i(h) + g_{i+1}(h)$. However, if layer i adjusts its computation based on the output of layer $i{+}1$, swapping the order changes the result. When layer $i{+}1$ is applied first, layer i outputs $g_i(h + g_{i+1}(h))$; in the reverse order, the result is $g_{i+1}(h + g_i(h))$. The more the two layers respond to each other's outputs, the larger this difference becomes.

This observation is the key to measuring coordination: **the degree to which the output changes when the order is swapped is precisely the degree to which the two layers depend on each other.** For independent layers, the difference is zero; for strongly cooperative layers, the difference is large.

2.2 Lie Bracket: The Mathematical Structure of Cooperation

Expanding this difference mathematically reveals its structure. A first-order expansion of the output difference between the two orderings yields:

$$\begin{aligned} f_i(f_{i+1}(h)) - f_{i+1}(f_i(h)) \\ = \underbrace{J_{g_i}(h)\, g_{i+1}(h) - J_{g_{i+1}}(h)\, g_i(h)}_{[g_i, g_{i+1}](h)} \\ + O(\|g\|^3) \end{aligned} \tag{1}$$

where $J_{g_i}(h) = \partial g_i / \partial h$ is the Jacobian matrix of layer i. This first-order term is known in differential geometry as the *Lie bracket* $[g_i, g_{i+1}]$, and each term has a clear interpretation:

- $J_{g_i}(h)\, g_{i+1}(h)$: how differently layer i responds when the output of layer $i{+}1$ perturbs its input
- $J_{g_{i+1}}(h)\, g_i(h)$: the converse—the response of layer $i{+}1$ to the output of layer i

The Lie bracket captures the asymmetry between these two responses—it grows when one side responds more strongly to the other, or when the response directions differ. This antisymmetry means that the Lie bracket captures not merely the magnitude of interaction, but the existence of a new computational direction that emerges only from the combination of the two layers. When the Lie bracket is the zero vector, the two layers operate independently; when it is nonzero, the result depends on which layer acts first—indicating a structure of mutual dependence.

In our experiments, $\|g_i\|/\|h\|$ ranges from 0.03 to 0.17 (§3.3), making the $O(\|g\|^3)$ term negligible. To empirically measure this difference, we swap the execution order of two adjacent layers and compute the output difference:

$$\mathrm{CD}(i, i{+}1) = \frac{\|f_i(f_{i+1}(h)) - f_{i+1}(f_i(h))\|_2}{\|h\|_2} \tag{2}$$

where $\|\cdot\|_2$ denotes the L2 norm (all norms hereafter are L2).

2.3 Inter-Layer Coordination: Separating the Scale Effect

However, CD alone is insufficient. Layer pairs with large updates can exhibit high CD regardless of whether they actually cooperate. From the first-order expansion of the Lie bracket, $[g_i, g_{i+1}]$ is linear in each of g_i and g_{i+1}, so CD grows proportionally to $O(\|g_i\| \cdot \|g_{i+1}\|)$. We remove this scale effect to isolate the proportion of cooperation relative to each layer's individual contribution:

$$\mathrm{Coord}(i, i{+}1) = \frac{\mathrm{CD}(i, i{+}1)}{(\|g_i\|/\|h\|) \cdot (\|g_{i+1}\|/\|h\|)} \tag{3}$$

Inter-layer coordination (Coord) measures "how large is the computation arising from mutual responsiveness, relative to each layer's individual contribution." For example, a coordination value of 6 means that the computation generated by the two layers responding to each other is 6 times the product of their individual contributions. Low values indicate that layers process computations independently, while high values indicate that the layers have evolved to perform computations through cooperation that exceed their individual limits.

2.4 Experimental Setup

Models and checkpoints. To track how coordination changes over the course of pretraining, we analyze 6 models across 3 architecture families with publicly available pretraining checkpoints: **Pythia** (Biderman et al., 2023) (GPT-NeoX, 70M–6.9B) 7 variants, **OLMo** (Groeneveld et al., 2024) (1B/7B), and **Amber** (LLaMA, 7B). For tracking training trajectories, we use Pythia 410M/1B/1.4B (6 checkpoints each: steps 0, 1K, 4K, 16K, 64K, 143K), OLMo-1B (7 checkpoints: steps 0, 1K, 10K, 50K, 100K, 500K, 1M), OLMo-7B (6 checkpoints: steps 1K–400K), and Amber-7B (4 checkpoints: ckpt 0–300).

Coordination computation. At each checkpoint, we compute CD for all adjacent layer pairs $(i, i{+}1)$. Each pair requires 4 single-layer forward passes: the output in normal order $f_{i+1}(f_i(h))$, the output in swapped order $f_i(f_{i+1}(h))$, and each layer's individual update $g_i(h)$ and $g_{i+1}(h)$. The total computational cost is equivalent to $\sim$4 full forward passes through the entire model.

Input texts. Coordination is computed as the average over 1,000 texts randomly sampled from WikiText-103 (Merity et al., 2016). To verify input robustness, we measured the pairwise Pearson correlation of coordination profiles across texts, confirming high stability with > 0.95 across all models.

Functional validation. To validate the functional significance of coordination structures, we select the block with the highest coordination (High-Coord) and the block with the lowest coordination (Low-Coord) for each of the 6 models, duplicate them, and measure the performance change across 6 downstream benchmarks (ARC-Easy, ARC-Challenge, HellaSwag, PIQA, WinoGrande, LAMBADA; all 0-shot). The block size is set to 2 layers.

3 Changes in Coordination During Pretraining

We apply the inter-layer coordination introduced in §2 at each stage of pretraining to track how inter-layer cooperative structures form.

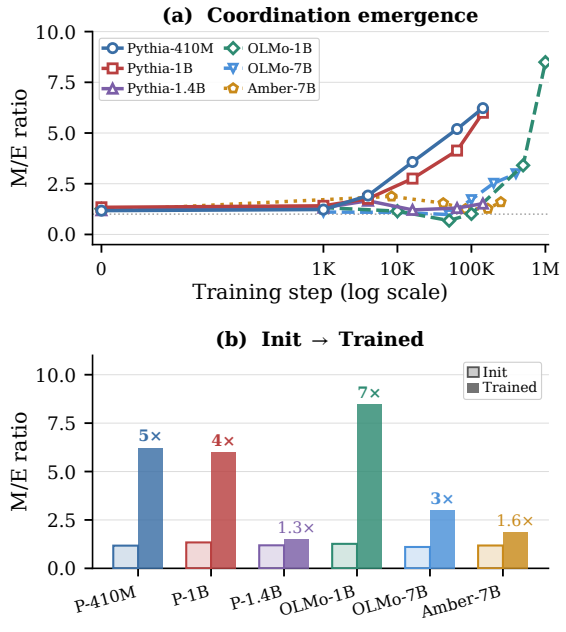


Figure 1: Changes in coordination during pretraining (averaged over 1,000 texts). (a) All 4 models start at M/E≈1× at initialization and develop concentrated coordination in middle layers after pretraining. (b) Final M/E ratio relative to initialization.

3.1 Emergence of Coordination

To verify whether pretraining actually builds inter-layer cooperation, we track the coordination of all adjacent layer pairs from the beginning to the end of pretraining for each model. To summarize the spatial distribution of coordination, we divide all layers into three equal regions—Early, Mid, and Late—and define the M/E ratio as the mean coordination of the Mid region divided by that of the Early region. M/E≈1 indicates that coordination is uniformly distributed, while M/E>1 indicates that coordination is concentrated in the middle layers.

Step	410M	1B	1.4B
0	1.17×	1.34×	1.19×
1,000	1.23×	1.41×	1.32×
4,000	1.92×	1.76×	1.66×
16,000	3.57×	2.75×	1.21×
64,000	5.20×	4.13×	1.30×
143,000	**6.23**×	**6.00**×	**1.52**×

Table 1: M/E ratio progression for three Pythia models (1,000 texts). Starting at ≈1× at initialization, coordination concentrates in middle layers as pretraining progresses.

Figure 1 and Table 1 present the results. Immediately after initialization, all models show M/E≈1×, with virtually no difference in coordination across layer positions. This indicates that randomly initialized layers are still operating independently of one another. However, as pretraining progresses, the pattern changes. Pythia-410M sees its M/E rise from 1.17× to 6.23×, and Pythia-1B from 1.34× to 6.00×, with middle-layer coordination becoming more than 6 times stronger than that of early layers. In Pythia, this increase occurs sharply after step 1K, suggesting that inter-layer cooperation begins to form in earnest at a specific point early in pretraining.

To confirm that this pattern is not limited to Pythia, we perform the same analysis on OLMo and Amber. Clear increases are confirmed in OLMo-1B (1.27×→8.49×) and OLMo-7B (1.10×→3.00×). The same direction is also confirmed in Amber-7B (1.18×→1.88×). Across all 6 models, the pattern is consistent: coordination is uniform at initialization and concentrates in the middle layers as pretraining progresses.

A directional analysis confirms that at initialization the two Lie bracket terms behave as random vectors (retention ratio $\approx 1/\sqrt{2}$), and pretraining partially aligns them while preserving substantial non-commutativity (retention ratio 0.59–0.67; Appendix A.2).

3.2 Decreasing Individual Contributions × Increasing Coordination

We analyze how the regions where coordination increases relate to existing metrics. Figure 2 presents the results for Pythia-1.4B. The middle layers (L8–15) have the minimum Block Influence (0.041) and maximum CKA (1.000)—existing metrics judge this region as "doing almost nothing."

However, coordination tells the opposite story. The coordination of the middle layers is 5.99, the highest compared to early layers (3.94) and late layers (5.42). The layers that individually produce the smallest changes generate the largest computation through cooperation with their neighbors. This inversion is confirmed in Pythia-410M (M/E 6.23×), Pythia-1B (6.00×), OLMo-1B (8.49×), and OLMo-7B (3.00×). Including Amber-7B (1.88×), the inversion is observed in 6/6 models, indicating a pattern

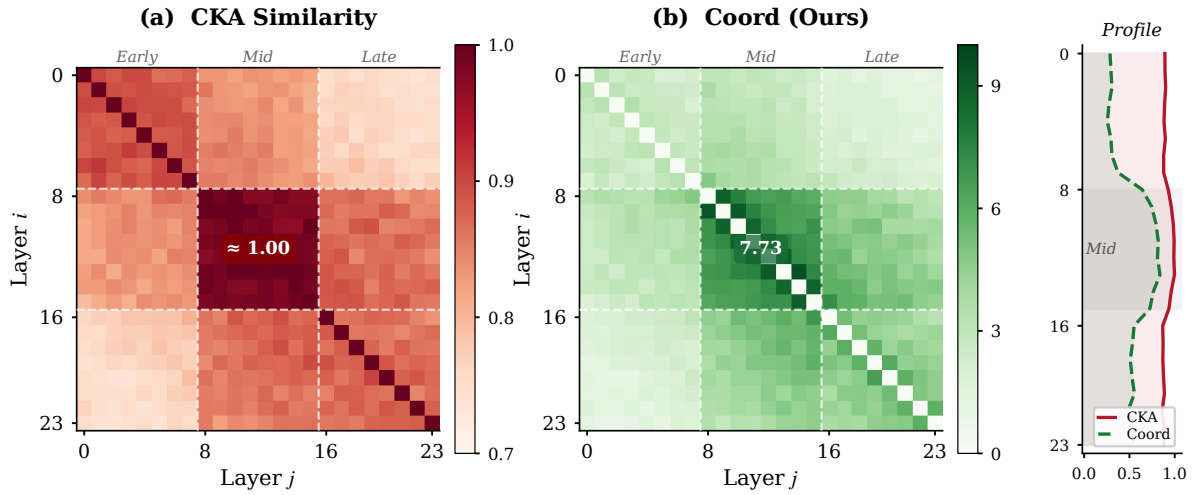


Figure 2: Comparison of CKA (a) and coordination (b) matrices (Pythia-1.4B). In CKA, the middle-layer block shows uniform similarity of ≈1.00, whereas in coordination, the same region shows the maximum values.

not limited to any specific architecture.

That this inversion is a product of pretraining rather than an architectural artifact is confirmed by the fact that at initialization, M/E≈1× and no inversion exists (Table 1). Pretraining does not increase each layer's individual contribution; instead, it reduces individual contributions while building inter-layer cooperation. The middle layers that appear to "do nothing" individually are in fact performing computations through cooperation with adjacent layers that would be impossible alone.

3.3 Mechanism of Coordination: Jacobian Amplification

Why is coordination maximal in the middle layers where individual contributions are smallest? To understand this, we analyze each layer's amplification ratio—the Jacobian norm divided by the output norm. The amplification ratio measures "this layer's output is small, but how sensitively does it respond to small changes in its input."

Table 2 presents the results. Middle layers have extremely small outputs ($\|g_i\|/\|h\| \approx 0.03$), yet their input sensitivity reaches up to 53 times their output magnitude. This connects directly to the Lie bracket mechanism defined in §2.2: in $J_{g_i} g_{i+1}$, when the Jacobian J_{g_i} is large, even a small output g_{i+1} from the adjacent layer is greatly amplified as it passes through layer i. In other words, middle layers adopt a strategy of "writing small but responding sensitively," which is why small individual contributions and large cooperative computations coexist.

Model	Early	Mid	Late	Mid/Early
Pythia-410M	1.27	16.46	5.94	13.0×
Pythia-1B	1.12	12.85	6.79	11.4×
Pythia-1.4B	3.93	8.71	4.65	2.2×
OLMo-1B	2.87	31.08	21.83	10.8×
OLMo-7B	19.03	53.12	39.46	2.8×
Amber-7B	1.87	3.76	1.90	2.0×

Table 2: Amplification ratios (Jacobian norm / output norm) for 6 models. The amplification ratio of middle layers is the highest across all models.

Notably, this responsiveness is bilateral: the magnitude ratio $\|A\|/\|B\|$ between the two directional terms of the Lie bracket remains within 0.89–1.04 across all models, indicating that coordination arises from mutual sensitivity rather than unidirectional dependency (Appendix A.1).

This amplification structure is also shaped by pretraining. Tracking the M/E ratio of amplification ratios across training checkpoints of Pythia-410M, 1B, and OLMo-1B, the learning trajectory of amplification M/E shows extremely high correlation with that of coordination M/E (Pythia-410M: r=0.997, Pythia-1B: r=0.995, OLMo-1B: r=0.998, all $p < 0.001$). In all three models, amplification and coordination first reach 2.0× at the same training step (Pythia: step 4K, OLMo: step 500K).

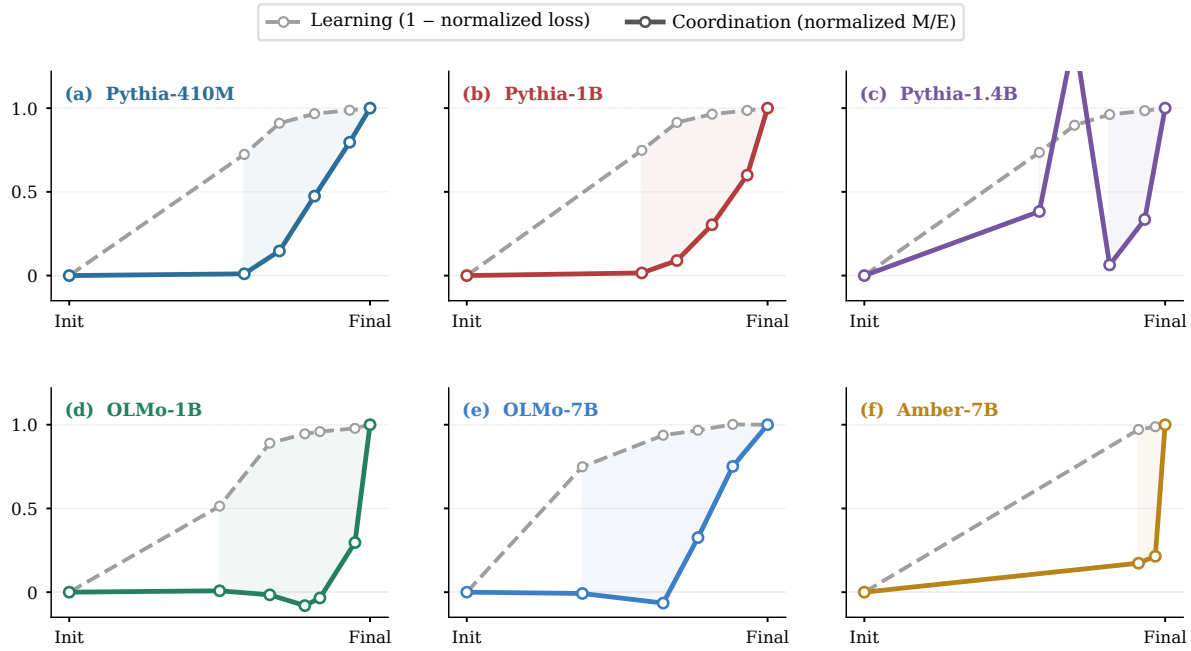


Figure 3: Comparison of training progress and coordination formation rates. For each model, the eval loss reduction rate (gray dashed line) and M/E ratio increase rate (colored solid line) are normalized to 0–1 relative to initialization and final values. The shaded region indicates the transition zone between the point where loss reduction begins to slow and the point where coordination begins to form in earnest.

A decomposition of the Lie bracket into attention and FFN components reveals that FFN accounts for 62–80% of the coordination response across all 6 models, identifying FFN as the primary driver of the "write small, respond sensitively" strategy (Appendix A.3).

3.4 Coordination Building Explains Diminishing Returns in Late Pretraining

It is well known that the pretraining loss curve exhibits a rapid initial decrease followed by a prolonged period of gradual improvement. However, what the model is *doing* internally during this latter phase has not been clearly explained. In this section, we show that this phase of diminishing returns temporally coincides with the building of inter-layer coordination.

At each checkpoint, we measure the eval loss (WikiText-103, 1,000 texts) and compare the loss reduction rate and M/E ratio increase rate on the same 0–1 normalized scale (Figure 3). Across all 6 models, the relationship between the two curves is consistent: coordination remains nearly stagnant during the phase of rapid loss decrease, and begins to rise substantially only after loss reduction slows. In Pythia-410M, the eval loss reaches approximately 75% of its total reduction by step 4K, but the M/E ratio has achieved only 15% of its final increase at the same point. This pattern is reproduced in Pythia-1B, OLMo-1B, and OLMo-7B.

These results suggest that the two phases of pretraining are qualitatively different processes. In the early phase, each layer independently acquires useful representations, rapidly reducing the loss. Once individual layers' representations have matured, the primary task of pretraining shifts to organizing these representations into inter-layer cooperative structures. From this perspective, the diminishing returns in late pretraining can be interpreted not as the model "ceasing to learn," but as the model performing a structurally different kind of learning—connecting individual components into cooperative circuits.

4 Functional Validation of Coordination Structure

In §3, we showed that pretraining systematically builds inter-layer coordination structures. However, does this coordination substantively contribute to model function, or is it merely a byproduct of pretraining? To answer this question, we duplicate layer blocks and analyze the

Model		ARC-E	ARC-C	HeSw	PIQA	WinoG	LAMBD	Avg	
P-410M	Base	45.7	24.2	40.6	67.1	53.3	51.4	47.1	3.7×
	High Δ	+0.5	+1.3	−1.9	−1.3	−0.3	−6.3	**−1.3**	
	Low Δ	−2.3	−1.1	−3.3	−2.7	−1.8	−18.3	−4.9	
P-1B	Base	49.0	27.0	47.2	69.4	53.5	56.2	50.4	5.1×
	High Δ	−1.5	−0.4	−2.7	−2.0	+0.2	−10.1	**−2.7**	
	Low Δ	−12.3	−4.6	−13.1	−8.5	−3.2	−42.0	−14.0	
P-1.4B	Base	53.8	28.6	52.0	71.1	57.4	61.5	54.1	4.8×
	High Δ	−2.8	+0.0	−2.7	−1.4	+0.2	−5.9	**−2.1**	
	Low Δ	−10.4	−3.2	−12.6	−6.4	−4.2	−24.4	−10.2	
OLMo-1B	Base	57.3	30.9	62.9	75.2	59.9	56.2	57.1	2.9×
	High Δ	−2.8	−1.4	−0.9	−0.7	−1.1	−12.5	**−3.2**	
	Low Δ	−10.0	−0.8	−3.9	−2.6	−3.2	−35.0	−9.2	
OLMo-7B	Base	74.4	43.6	77.8	80.3	67.4	71.6	69.2	7.9×
	High Δ	−3.0	−2.2	−0.3	−0.7	−1.8	−1.5	**−1.6**	
	Low Δ	−9.8	−4.2	−13.2	−4.2	−11.2	−32.3	−12.5	
Amber-7B	Base	63.7	35.1	72.5	78.6	64.7	67.9	63.7	2.7×
	High Δ	−0.7	−0.5	+0.1	−0.3	−2.8	−3.5	**−1.3**	
	Low Δ	−2.6	−3.5	−6.7	−2.0	−2.9	−3.1	−3.5	

Table 3: Block duplication experiment: performance change (Δ%p) across 6 benchmarks. We duplicate High-Coord/Low-Coord blocks and measure the change in 0-shot accuracy. Block size is 2L. Across all 6/6 models, High-Coord blocks are 2.7–7.9× more robust.

resulting performance changes on downstream benchmarks.

High-coordination blocks are blocks in which adjacent layers are precisely tuned to each other's outputs. Duplicating such a block—i.e., passing through the same computation one additional time—merely repeats the tuned structure, so its impact on model function should be limited. In contrast, low-coordination blocks consist of relatively independent layers, so duplication may introduce redundant computation that disrupts the overall information flow. To test this prediction, for each of the 6 models we select the 2-layer block with the highest coordination (High-Coord) and the 2-layer block with the lowest coordination (Low-Coord), duplicate them, and measure the performance change across 6 benchmarks (ARC-Easy, ARC-Challenge, HellaSwag, PIQA, WinoGrande, LAMBADA; all 0-shot).

Table 3 presents the results. Across all 6 models, duplicating High-Coord blocks is 2.7–7.9× more robust than duplicating Low-Coord blocks. When blocks with high coordination—blocks where inter-layer cooperation is strong—are duplicated, benchmark performance is consistently better preserved. This pattern holds at the individual benchmark level as well: out of 6 models × 6 benchmarks = 36 comparisons, the performance drop from High-Coord duplication is smaller than that from Low-Coord duplication in 33 cases.

A benchmark-level analysis provides clues about the functional role of this coordination structure. The benchmarks most sensitive to coordination disruption are LAMBADA (gap 19.2%p) and HellaSwag (7.4%p), both of which require the gradual integration of long-range context. In contrast, PIQA (3.3%p) and WinoGrande (3.5%p), which require relatively local judgments, are less sensitive. This supports our observation that middle layers perform cooperative refinement rather than independent computation, and is consistent with prior findings that consecutive layers in residual networks perform iterative refinement (Jastrzebski et al., 2018; Greff et al., 2017). The observation from Tuned Lens (Belrose et al., 2023) that "predictions converge monotonically across layers" further supports the view that this refinement is achieved through inter-layer coordination. Moreover, given the finding that factual recall follows a multi-stage pipeline across early, middle, and late layers (Geva et al., 2023), it is noteworthy that the middle layers where coordination peaks correspond precisely to the "relation integration" stage. The heightened sensitivity of LAMBADA can be attributed to the fact that long-range context prediction depends on the entire multi-stage refinement

chain.

These results demonstrate that the coordination structure observed in §3 is not a byproduct of pretraining but a computational structure on which the model genuinely relies.

5 Related Work

Pretraining dynamics. Prior work has identified structured reorganization during pretraining, including the emergence of induction heads (Olsson et al., 2022; Singh et al., 2024), knowledge entropy decay (Kim et al., 2025), and power-law knowledge acquisition (Chang et al., 2024). These focus on specific circuits; we track the emergence of a general inter-layer coordination structure.

Layer analysis and redundancy. BI (Men et al., 2024) and CKA (Kornblith et al., 2019) evaluate layers individually, leading to claims that middle layers are redundant (Men et al., 2024; Gromov et al., 2024). Yet Skean et al. (2025) showed these layers yield the strongest representations, and Sharma et al. (2024) found they mix removable noise with essential signals. Our coordination metric resolves this tension: the "redundant" middle layers are the locus of inter-layer cooperation invisible to single-layer metrics.

Inter-layer relationships and information flow. Veit et al. (2016) suggested layer independence, while circuit analyses (Elhage et al., 2021; Wang et al., 2023), iterative refinement studies (Jastrzebski et al., 2018; Belrose et al., 2023), cross-layer superposition (Lindsey et al., 2024), and multi-step circuit tracing (Batson et al., 2025) demonstrated inter-layer coupling. We add a developmental axis: layers are independent at initialization but build cooperative structures through pretraining.

Model compression and mathematical frameworks. LayerDrop (Fan et al., 2020), depth-adaptive inference (Elbayad et al., 2020), and layer permutation analyses (Sun et al., 2025; Lad et al., 2024) focus on individual layer removability; our metric predicts robustness to block-level perturbations. Methodologically, Neural ODEs (Chen et al., 2018) and interacting particle models (Geshkovski et al., 2025) treat depth as continuous, and Walker et al. (2024) computed Lie brackets of neural network vector fields. We apply this geometric perspective to measure inter-layer coordination in pretrained transformers.

6 Discussion and Conclusion

What pretraining builds. Existing tools measure the individual properties of each layer—a form of hardware inspection. Coordination measures the cooperative relationships between components. Our results suggest that pretraining is not merely a process of installing functions into hardware, but of installing cooperative circuits between them. Analogous to how Olsson et al. (2022) demonstrated the emergence of a specific circuit (induction heads), this work shows that a general layer-level coordination structure emerges through pretraining.

Reinterpreting depth. Probing studies report that representations are richest in the middle layers (Belinkov, 2022), yet BI says they "do almost nothing." This paradox is resolved by our findings: middle layers perform individually small transformations, but through cooperation with adjacent layers, they carry out computations that would be impossible alone. The combination of small updates and high Jacobians simultaneously preserves gradient flow and generates rich computational directions via large Lie brackets—a strategy of "writing small but responding sensitively" that balances stability and expressiveness. Depth can thus be reinterpreted not as "more sequential transformations" but as "room for more precise inter-layer cooperation."

Summary. Taken together, our Lie bracket-based coordination metric reveals that pretraining systematically builds inter-layer cooperation—concentrated in the very layers existing metrics dismiss as redundant. This cooperation is mechanistically grounded in Jacobian amplification, functionally validated by robustness to structural perturbation, and temporally aligned with the diminishing-returns phase of training. These findings reframe pretrained transformers not as stacks of independently functioning layers, but as systems whose power arises from the cooperation installed between them.

Limitations

Model scale. All models analyzed in this study range from 70M to 7B parameters. While the coordination patterns we identify—uniform initialization, systematic concentration in middle layers, and the inversion with BI/CKA—are consistent across this range and across three distinct architecture families, we have not verified whether these findings extend to models at the 13B+ scale, where additional phenomena such as deeper layer specialization may arise. We note, however, that the trends observed from 70M to 7B show no sign of diminishing, suggesting that inter-layer coordination is likely a general property of pretrained transformers rather than a scale-specific artifact.

Mechanistic granularity. Our coordination metric operates at the level of entire transformer blocks, quantifying how much two adjacent layers depend on each other without decomposing *which internal components* (e.g., specific attention heads or FFN neurons) drive the coordination. Computing the full Jacobian for fine-grained attribution is prohibitively expensive at scale; our swap-based measurement sidesteps this cost but trades off mechanistic detail. Combining coordination with circuit-level tools such as path patching (Wang et al., 2023) or cross-layer transcoders (Batson et al., 2025) to identify the specific sub-components responsible for high coordination is a promising direction for future work.

Use of AI Assistants

This work utilized AI assistants in the following capacities:

- **Translation:** Translating draft text for manuscript preparation.
- **Code Development:** Assisting with debugging and refining experimental code for coordination computation and evaluation pipelines.

All research conceptualization, experimental design, data analysis, and scientific conclusions are solely the work of the authors. The authors take full responsibility for the content of this paper.

Acknowledgments

This work was supported by the Ministry of Science and ICT and the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2026-25498341).

References

Joshua Batson, Adly Templeton, et al. 2025. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

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

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. *International Conference on Machine Learning*, pages 2397–2430.

Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. 2024. How do large language models acquire factual knowledge during pretraining? In *Advances in Neural Information Processing Systems*, volume 37.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *International Conference on Learning Representations*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. 2025. A mathematical perspective on transformers. *Bulletin of the*

American Mathematical Society, 62(3):427–479. ArXiv:2312.10794.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Klaus Greff, Rupesh Kumar Srivastava, and Jürgen Schmidhuber. 2017. Highway and residual networks learn unrolled iterative estimation. In *International Conference on Learning Representations*.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. OLMo: Accelerating the science of language models. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Stanislaw Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. 2018. Residual connections encourage iterative inference. In *International Conference on Learning Representations*.

Jiyeon Kim, Hyunji Lee, Hyowon Cho, Joel Jang, Hyeonbin Hwang, Seungpil Won, Youbin Ahn, Dohaeng Lee, and Minjoon Seo. 2025. Knowledge entropy decay during language model pretraining hinders new knowledge acquisition. In *International Conference on Learning Representations*. Oral. arXiv:2410.01380.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. *International Conference on Machine Learning*, pages 3519–3529.

Vedang Lad, David Mahan, Anh Totti Pham, Khaled Saab, Jing Yu Tian, et al. 2024. The remarkable robustness of llms: Stages of inference? *arXiv preprint arXiv:2406.19384*.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Tom Conerly, Joshua Batson, and Chris Olah. 2024. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. 2022. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*.

Pratyusha Sharma, Jordan T. Ash, and Dipendra Misra. 2024. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. In *International Conference on Learning Representations*. ArXiv:2312.13558.

Aaditya K. Singh, Ted Moskovitz, Felix Hill, Stephanie C. Y. Chan, and Andrew M. Saxe. 2024. What needs to go right for an induction head? A mechanistic study of in-context learning circuits and their formation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pages 45637–45662.

Oscar Skean et al. 2025. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*.

Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. 2025. Transformer layers as painters. In *Proceedings of the AAAI Conference on Artificial Intelligence*. ArXiv:2407.09298.

Andreas Veit, Michael J Wilber, and Serge Belongie. 2016. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, volume 29.

Benjamin Walker, Andrew D. McLeod, Tiexin Qin, Yichuan Cheng, Haoliang Li, and Terry Lyons. 2024. Log neural controlled differential equations: The Lie brackets make a difference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pages 49822–49844.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*.

A Decomposing the Lie Bracket: Directional Analysis of Inter-Layer Coordination

In the main text, we quantified inter-layer coordination by measuring the output difference when two adjacent layers' execution order is swapped (§2). However, this scalar measurement aggregates two distinct sources of interaction: layer i's response to layer $i{+}1$'s output, and layer $i{+}1$'s response to layer i's output. The aggregated coordination metric does not reveal the internal structure of this interaction—whether the two responses are symmetric in magnitude, how their directions relate, or which architectural component drives them.

The first-order expansion of the Lie bracket provides a natural decomposition. As derived in §2.2, the output difference under order swap decomposes into two terms:

$$[g_i, g_{i+1}](h) = \underbrace{J_{g_i}(h)\, g_{i+1}(h)}_{\text{Term A: layer } i\text{'s response}} - \underbrace{J_{g_{i+1}}(h)\, g_i(h)}_{\text{Term B: layer } i{+}1\text{'s response}} \quad (4)$$

where $J_{g_i}(h) = \partial g_i / \partial h$ is the Jacobian of layer i at input h. Term A measures how much layer i's output changes when layer $i{+}1$'s output perturbs its input; Term B measures the converse. Each term can be estimated without explicit Jacobian computation via forward passes alone:

$$\text{Term A} \approx g_i(h + g_{i+1}(h)) - g_i(h) \quad (5)$$

$$\text{Term B} \approx g_{i+1}(h + g_i(h)) - g_{i+1}(h) \quad (6)$$

This requires only two additional forward passes per layer pair beyond the infrastructure used for coordination computation in §2.4. Computing the full Jacobian would cost $O(d^2)$; this method recovers the Jacobian's action on the most operationally relevant direction—the actual output of the neighboring layer—at $O(d)$ cost.

We derive two diagnostics from these terms: (1) **directional symmetry** $\|A\|/\|B\|$, measuring whether one layer dominates the interaction, and (2) **retention ratio** $\|A - B\|/(\|A\| + \|B\|)$, measuring how much of the total response survives as the Lie bracket rather than canceling out. We apply this decomposition to the final checkpoints of all 6 models and to intermediate training checkpoints.

A.1 Symmetric Magnitudes: Both Layers Respond Equally

We first ask whether inter-layer coordination is driven by one direction of interaction or balanced across both. For each adjacent layer pair $(i, i{+}1)$, we compute the magnitude ratio $\|A\|/\|B\|$, averaged over the middle-layer region and 1,000 WikiText-103 texts. A ratio of 1.0 indicates that both layers respond with equal magnitude to each other's outputs; substantial deviation from 1.0 would indicate that coordination is dominated by one layer's sensitivity.

Figure 4(a) shows $\|A\|/\|B\|$ across the training trajectory for all 6 models. The ratio remains within 0.89–1.04 across all models from initialization through training completion. This near-unity ratio is maintained despite the substantial increase in overall coordination magnitude reported in §3.1 (e.g., M/E ratio from 1.17× to 6.23× in Pythia-410M). The result indicates that pretraining does not create a unidirectional dependency—where one layer becomes a "sender" and the other a "receiver"—but instead builds **mutual responsiveness of balanced magnitude**. Both layers in a coordinated pair increase their sensitivity to each other's outputs in tandem.

This symmetry is consistent with the Jacobian amplification mechanism described in §3.3. Since middle layers share similarly high amplification ratios (Table 2), adjacent layers in this region would be expected to respond to each other's perturbations with comparable magnitude, which is precisely what we observe.

A.2 Divergent Directions: Responses That Do Not Cancel

The near-unity magnitude ratio in §A.1 raises a question: if $\|A\| \approx \|B\|$, the Lie bracket $\|A - B\|$ could be anywhere from 0 (if A and B point in the same direction, fully canceling) to $\|A\| + \|B\|$ (if they point in opposite directions, fully reinforcing). The *direction* of the two responses, not just their magnitude, determines how much coordination survives in the Lie bracket.

To quantify this, we define the **retention**

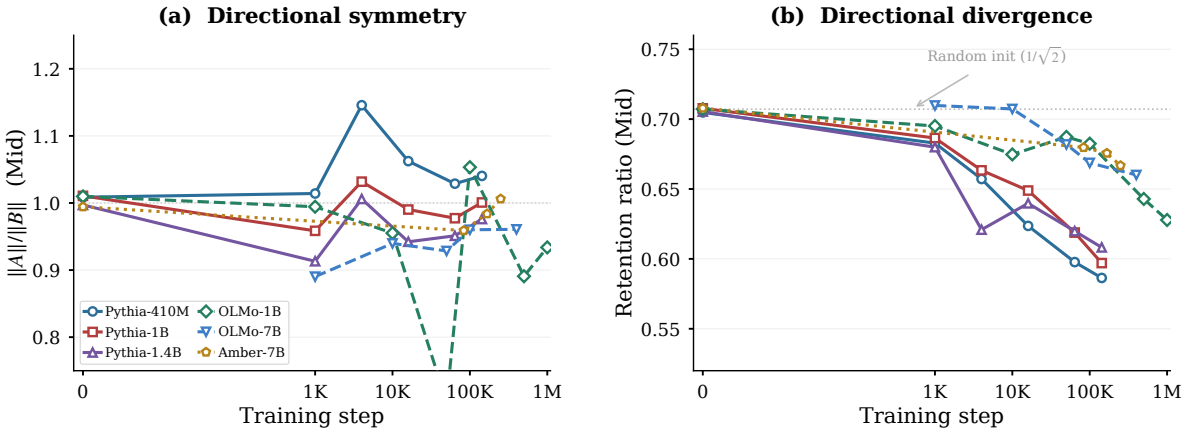


Figure 4: Lie bracket decomposition across pretraining. (a) Directional symmetry $\|A\|/\|B\|$ remains ≈1.0 from initialization through training completion in all 6 models. (b) Retention ratio $\|A-B\|/(\|A\|+\|B\|)$ starts at $1/\sqrt{2} \approx 0.71$ (the theoretical expectation for random high-dimensional vectors) and decreases to 0.59–0.67 as training progresses.

ratio:

$$\text{Retention ratio} = \frac{\|A - B\|}{\|A\| + \|B\|} \tag{7}$$

A retention ratio of 0 means the two terms are parallel and fully cancel, leaving zero Lie bracket—the execution order is irrelevant. A ratio of 1.0 means no cancellation occurs and the full combined magnitude is preserved in the Lie bracket. For two random vectors in $\mathbb{R}^d$ with $d \gg 1$, the expected retention ratio converges to $1/\sqrt{2} \approx 0.707$.

Figure 4(b) presents the results. Two findings emerge. First, at initialization, the retention ratio is ≈ 0.71 across all models, matching the theoretical expectation for random high-dimensional vectors. This confirms that the two terms are unstructured at initialization—consistent with the finding in §3.1 that coordination is uniform and minimal before training.

Second, pretraining reduces the retention ratio to 0.59–0.67. This indicates that the two response directions become *partially aligned* through training. However, the ratio remains well above zero, meaning that approximately 60% of the combined response magnitude survives as the Lie bracket. Although training introduces some directional alignment between the two terms, the responses remain **substantially non-parallel**, preserving the non-commutativity that constitutes coordination.

Combining §A.1 and §A.2, we can characterize the coordination structure as follows: two adjacent middle layers respond to each other with *similar magnitudes* but in *substantially different directions*. It is this directional divergence—not a magnitude asymmetry—that makes the execution order meaningful and gives rise to the coordination measured in the main text. Furthermore, what pretraining changes and what it does not are now clear: the bilateral symmetry is an architectural prior present from initialization, and pretraining increases coordination magnitude within this symmetric framework through Jacobian amplification.

A.3 FFN Dominates Inter-Layer Coordination

Each transformer layer's update $g_i(h)$ is the sum of an attention component and an FFN component. To identify which sub-module drives inter-layer coordination, we decompose Term A and Term B by measuring each component's response to the neighboring layer's perturbation:

$$J_{g_i}\,\delta = \underbrace{[\text{attn}_i(h+\delta) - \text{attn}_i(h)]}_{J_{\text{attn}}\,\delta} + \underbrace{[\text{ffn}_i(h+\delta) - \text{ffn}_i(h)]}_{J_{\text{ffn}}\,\delta} \tag{8}$$

where $\delta = g_{i+1}(h)$. The percentage contribution of each component is $\|\text{Term}^{\text{comp}}\|/(\|\text{Term}^{\text{attn}}\| + \|\text{Term}^{\text{ffn}}\|)$.

Table 4 presents the results. Across all 6 models, FFN accounts for 62–80% of both Term A and Term B. The pattern is consistent at two

Model	Residual	A (Attn)	A (FFN)	B (Attn)	B (FFN)
P-410M	Parallel	24%	**76%**	21%	**79%**
P-1B	Parallel	22%	**78%**	20%	**80%**
P-1.4B	Parallel	23%	**77%**	21%	**79%**
OLMo-1B	Sequential	35%	**65%**	37%	**63%**
OLMo-7B	Sequential	35%	**65%**	37%	**63%**
Amber-7B	Sequential	35%	**65%**	38%	**62%**

Table 4: Attention vs. FFN contribution to each Lie bracket term in the middle layers, averaged over 1,000 texts. FFN dominates in all 6 models (±2%p within each architecture type).

levels: (1) within each model, the FFN fraction for Term A and Term B is nearly identical (±2%p), and (2) within each architecture type, model-to-model variation is minimal.

The systematic difference between architecture types has a structural explanation. In parallel residual architectures (Pythia/GPT-NeoX), attention and FFN both receive the same pre-update input h, and the FFN's larger parameter count and stronger nonlinearity (GELU) lead to higher sensitivity, concentrating the response (∼78%). In sequential residual architectures (OLMo, Amber/LLaMA), FFN receives the post-attention residual $h + g^{\text{attn}}(h)$ as input, so a perturbation δ affects both attention directly and FFN indirectly through the modified attention output. This gives attention a relatively larger direct share (∼35% vs ∼22%).

This result refines the Jacobian amplification mechanism identified in §3.3: the high input sensitivity of middle layers is primarily a property of their **FFN sub-modules**. The FFN's role as the dominant driver of inter-layer coordination is consistent with prior findings that FFN layers serve as the primary site of knowledge storage and transformation (Geva et al., 2023), and suggests that the "write small, respond sensitively" strategy identified in the main text is largely an FFN-level phenomenon.