# Transformer Layers as Painters

**Qi Sun[*2,3], Marc Pickett[*†1], Aakash Kumar Nain[1], Llion Jones[2]**

[1]Emergence AI
[2]Sakana AI, Japan
[3]Institute of Science Tokyo, Japan
{mpickett,anain}@emergence.ai, {qisun, llion}@sakana.ai

**Abstract**

Despite their nearly universal adoption for large language models, the internal workings of transformers are not well understood. We aim to better understand the impact of removing or reorganizing information throughout the layers of a pretrained transformer. Such an understanding could both yield better usage of existing models as well as to make architectural improvements to produce new variants. We present a series of empirical studies on frozen models that show that the lower and final layers of pretrained transformers differ from middle layers, but that middle layers have a surprising amount of uniformity. We further show that some classes of problems have robustness to skipping layers, running the layers in an order different from how they were trained, or running the layers in parallel. Our observations suggest that even frozen pretrained models may gracefully trade accuracy for latency by skipping layers or running layers in parallel.

**Code** — https://github.com/floatingbigcat/transformer_layers_as_painters

## 1 Introduction

The scale of transformer-based Large Language Models (LLMs), in the billions of parameters, makes it difficult to directly understand the models' behaviour after training. At the same time, each layer of a pretrained transformer has an identical architecture as the other layers, with the only difference being a layer's position in the hierarchy, and the values of the layer's parameters (Vaswani et al. 2017).

We find it helpful to think of the middle layers of a transformer by making an analogy to an assembly line of painters. The canvas (input) is passed along a series of painters. Some painters specialize in birds, while others are better at painting wheels. Each painter receives the canvas from the painter below her, then she decides whether to add a few strokes to the painting or just pass it along to the painter above her (using the residual connections). In this analogy, each painter uses the same "vocabulary" for understanding paintings, so that a painter may receive the painting from a painter earlier in the assembly line without catastrophe. The painters may also be reordered without complete catastrophe (even if parts of the background get painted *after* foreground objects, occluding them), and the painters may even all add their strokes at the same time (in parallel).

This analogy isn't meant to be a rigorous theory, but rather a tool for thinking about a transformer's layers. Inspired by this analogy, we test how well some hypotheses hold. In this paper we perform experiments that help address the following questions:

1. Do layers use the same representation space? (§3.1)
2. Are all the layers necessary? (§3.2)
3. Are middle layers all doing the same function? (§3.3)
4. Does the layer order matter? (§3.4)
5. Can we run the layers in parallel? (§3.5)
6. Does order matter for some tasks more than others? (§3.6)
7. Does looping help parallelized layers? (§3.7)
8. Which variants harm performance the least? (§3.8)

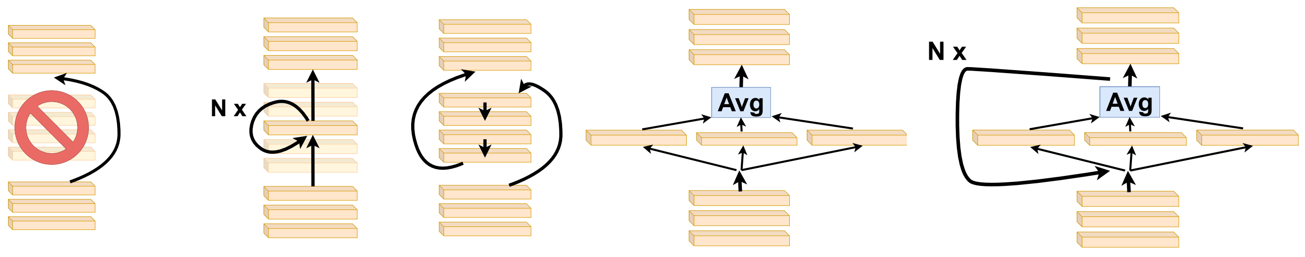
To answer these questions we perform a series of experiments on *pretrained* LLMs. These include experimenting with variations on the standard transformer execution strategy, and measuring the impact of these variations on the models' performance across a variety of benchmarks for both decoder-only (Llama) and encoder-only (BERT) models. Note that our experiments never involve finetuning or otherwise adjusting the models' parameters (with the caveat that the GLUE evaluation standard procedure includes a finetuning step for our BERT-Large model)

## 2 Models and Benchmarks

Our experiments are primarily on two transformer models: Llama2 (Touvron et al. 2023), and on BERT-Large (Devlin et al. 2019). (However, we also include results for Mistral-7B (Jiang et al. 2023) and Pythia-6.9B (Biderman et al. 2023a) in Appendix A.5 that support the generalization of our results.) Llama2 is *decoder-only*. We focus on Llama2-7B, which has 7 billion parameters and 32 layers (each layer having 202 million parameters), but also include some scaling experiments with the 13B (40 layers) and 70B (80 layers) models. BERT is *encoder-only* with 24 layers and 340 million parameters. We used the standard pretrained checkpoints for these models. In all our experiments the models are frozen: we never modified the parameters of these models through fine-tuning

[*]These authors contributed equally.
[†]Corresponding author.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



(a) Skip (b) Middle Repeat (c) Reverse (d) Parallel (e) Looped Parallel

Figure 1: Different execution strategies.

or other methods, with the exception of the BERT evaluation, which includes a standard fine-tuning step.

We used standard benchmarks for both *decoder-only* LLMs (for Llama2) and for *encoder-only* LLMs (for BERT). For Llama2, we use **ARC** (science exam questions) (Clark et al. 2018), **HellaSwag** (commonsense) (Zellers et al. 2019), **GSM8K** (Math Word Problems) (Cobbe et al. 2021), **WinoGrande** (Winograd Schema Challenge) (Sakaguchi et al. 2019), and **LAMBADA** (word prediction) (Paperno et al. 2016). This last, LAMBADA, measures perplexity and is closest to the raw token-prediction used during training. For Llama2, we include the *normalized median* of the benchmarks, where we scale each benchmark with 0 being the performance of random (or max-class) guessing and 1 being the performance of the full Llama2 model. For BERT, we used tasks from the GLUE benchmark (Wang et al. 2018) and followed their evaluation protocol, including reporting the *unnormalized average* of the benchmarks. Note that standard BERT evaluation includes a fine-tuning step (Devlin et al. 2019), so our BERT model has a chance to adapt to the new configuration. Therefore, we also include results from an evaluation where an additional output layer can adapt, but the model itself is frozen. These results are in Appendix A.9, and more details of the GLUE benchmark are given in Appendix A.8.

## 3 Experiments

The original motivation behind our experiments came from the question of whether multiple layers could be somehow be merged into a single (possibly larger) layer. (Such merging could potentially be automated (Akiba et al. 2024).) We hypothesized, perhaps because of the use of residual connections during training, that the middle layers of a neural network may use a common representation space. (This is not the case for standard multi-layer perceptrons, where there is nothing to encourage a common representation or permutational consistency across layers.) The possibility of layers sharing a common representation has downstream implications for conditional computation (e.g. (Pagliardini et al. 2024)) or for dynamically inserting new knowledge into pretrained transformer models.

### 3.1 Do Layers "Speak the Same Language"?

To answer whether different layers have a shared representation space, we test whether transformers are robust to skipping specific layers or switching the order of neighboring layers. For example, in Llama2-7B, layer 6 normally expects the output from layer 5. Would layer 6 behave catastrophically if it were given layer 4's output instead? In Figure **??**, we see that, with the important exception of the first and last few layers, Llama2-7B's layers are fairly robust to skipping or even switching layers (e.g., feeding layer 4's output to layer 6, then sending layer 6's output to layer 5, then to layer 7).

This experiment would suggest that the middle layers 1. share a representation space and 2. have a separate representation space from the "outer" (first and last few) layers. To further test this hypothesis, following previous work (Friedman et al. 2023; Kornblith et al. 2019; Simoulin and Crabbé 2021; Godey, Éric de la Clergerie, and Sagot 2024; Xue et al. 2023), we measured the average cosine similarity between the activations of hidden states of different layers of our models (Llama2-7B, Llama2-13B, and BERT-Large) across our benchmarks. In Figure 3, we show that this consistency holds among all the middle layers. For example, the activation in the fourth layer from the bottom has a high similarity to the fourth layer from the top. For the 40 layers of Llama2-13B, we see that the layers form four or five distinct similarity groups: Layer 0, layers 1-3, the middle layers, then the final layer or two.

This suggests that the model may have three distinct representation spaces for the "beginning", "middle", and "ending" layers. Note that in the 13B model, the number of "beginning layers" is 3 while the 7b is 2, the "ending layers" is 1 or 2 and 7b is clearly 2. So the number of "beginning layers" seems to grow as the total number of layers increases. (In Appendix A.3 we further show that these three classes are consistent across different model scales, with the beginning and middle layers growing proportionally to the total number of layers.) Also note that a high cosine similarity *may* suggest a shared representation space, but a low similarity is more indicative that the spaces are *not* shared. However, the fact that the matrix for Llama2-7B in Figure 3 aligns neatly with the performance shown in Figure **??** is stronger evidence that the *semantics* of the representation space is actually shared, at least for the middle layers. Based on this, we answer this subsection's question with:

**Yes, the middle layers seem to share a common representation space.**

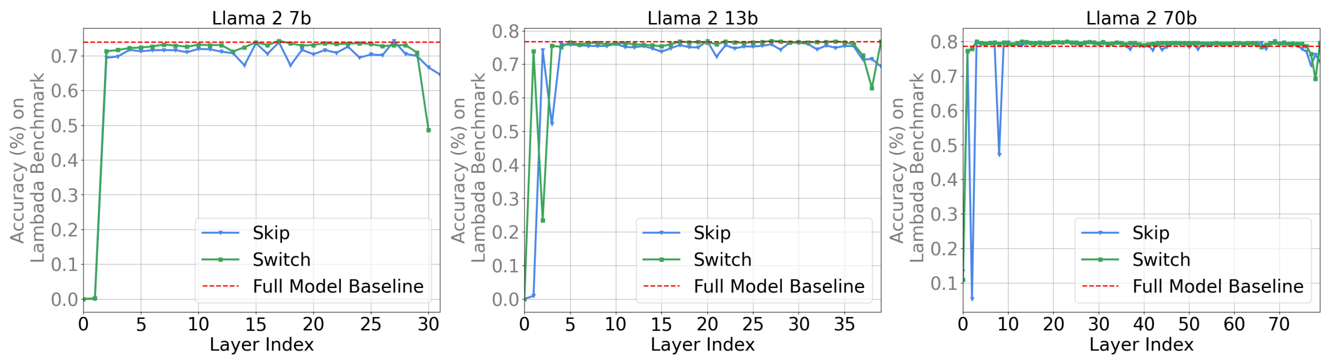


Figure 2: Results for Open-LAMBADA from *skipping* layer $N$ (blue), and from *switching* layer $N$ with $N+1$ (green) of Llama2-7B. Skipping early layers has a catastrophic effect, while the model is much more robust to skipping middle layers.

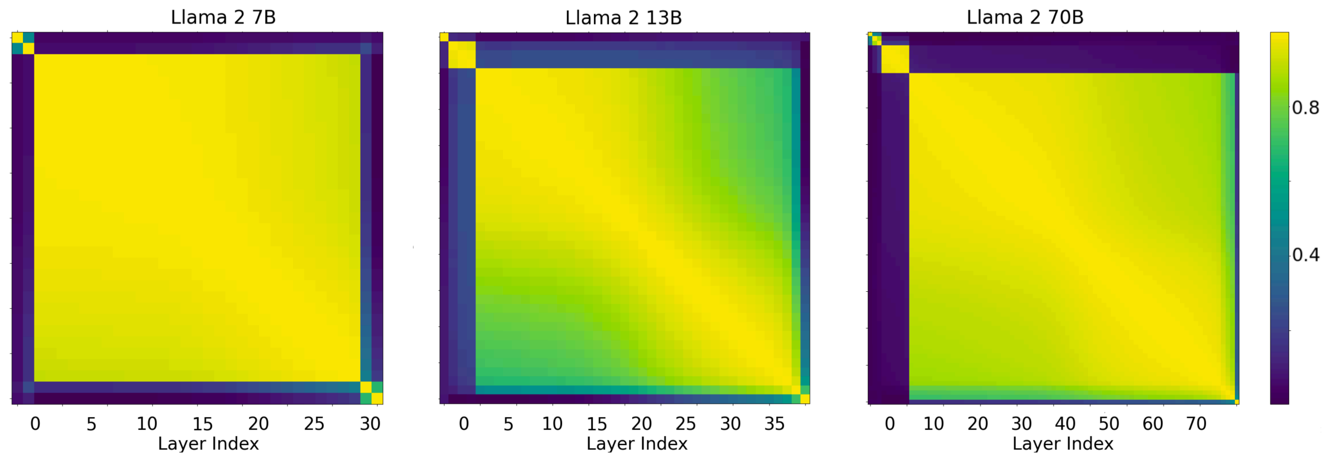


Figure 3: Avg. cosine similarity between the hidden states of all 32 layers of Llama2-7B (top) and all 40 layers of Llama2-13B.

## 3.2 Are All the Layers Necessary?

To further test whether the reorientation space for middle layers is truly shared (in addition to having close cosine similarity), we experiment with skipping layers. That is, we send the output of the $N$th layer directly into the input of layer $N+M$ (where $M>1$), thereby "skipping" $M-1$ layers, as illustrated in Figure 1a. Recall that we perform no fine-tuning during our experiments. Our experiments are to see if layer $N+M$ can make sense of activations from layer $N$, though it was trained only on inputs from layer $N+M-1$. For this (and related) experiments, we execute the first and last $N-1$ layers as normal, skipping (or later modifying) layers $N+1$ through $T-N$, where $T$ is the total number of layers in the model. Figure 4 shows that performance for many of our benchmarks has graceful degradation for both Llama2-7B and BERT-Large. (Note that the number of layers skipped is inversely proportional to $N$, so the plot goes from few skipped layers to many skipped layers when read from left to right.) This result suggests that the answer to whether all the layers are necessary is:

**No, at least a few middle layers can be dropped without catastrophic failure**.

In Appendix A.1, we analyze the layer skipping behavior across model sizes, revealing a surprisingly uniform pattern in the importance of middle layer partitions. Furthermore, in Appendix A.2, we demonstrate that fine-tuning can enhance performance when skipping fewer layers but becomes harmful when skipping too many.

## 3.3 Are Middle Layers All Doing the Same Thing?

If the middle layers share a common representation space, does this mean that these layers are redundant? To test this, we reran the "Skip" experiments from the previous subsection, but instead of skipping the middle layers, we replaced their weights with those of the center layer, effectively looping on this layer for $T-2N+1$ times, where $T$ is the total number of layers (32 for Llama2-7B, 24 for BERT-Large). (See illustration in Figure 1b.)

In Figure 5, we see that the benchmarks quickly decay as the number of replaced layers increases, and Figure 11 shows that this variation is the most catastrophic of all we tried, significantly worse than just skipping layers[1]. Therefore, we answer our question with:

**No, sharing weights among middle layers is catastrophic, indicating that the middle layers are performing different functions.**

[1]In Appendix A.4, we further explore why skipping is better than recycling the center-most layer.

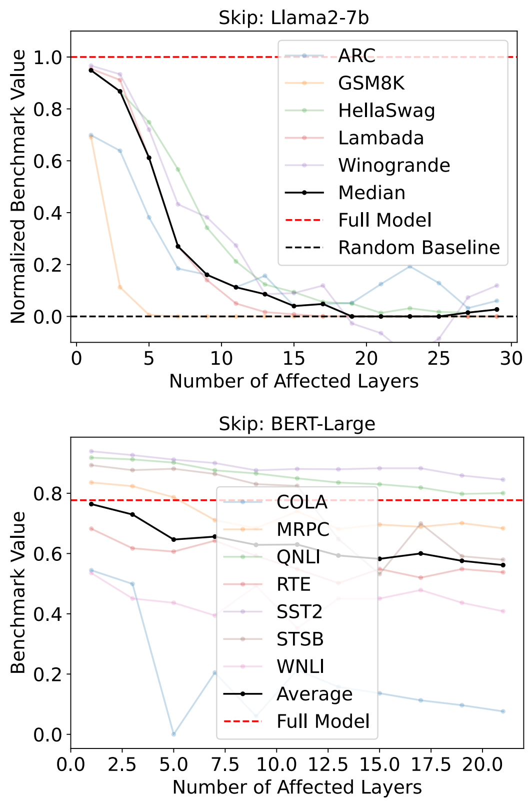


Figure 4: Top: Skipping layers N to 32-N for Llama2-7B, normalized per benchmark (median). Bottom: Skipping layers N to 24-N for BERT, with unnormalized average.

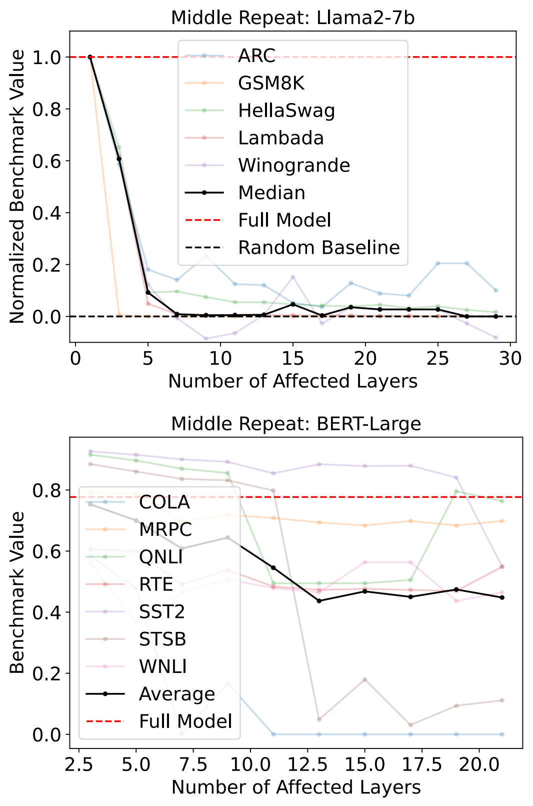


Figure 5: Replacing $M$ middle layers with the center layer (16 for Llama, 12 for BERT) for Llama2-7B (top, normalized benchmarks). and BERT (unnormalized average).

### 3.4 Does the Layer Order Matter?

The previous experiments suggest that middle layers share a representation space but perform different operations on this space. Another question is how much the order of these function matters. We performed two sets of experiments to test this. First, is running the middle layers in reverse order from how they were trained[2]. Specifically, we take the output of layer $T - N$ and send it into the input of $T - N - 1$, then the output of this layer into $T - N - 2$ and so on down to layer $N$, then send the output of this layer to the last $T - N$ layers. (See Figure 1c.) In the second variation we ran the middle layers in a random order (and averaged the results over 10 seeds).

The results for Reversed and Random Order are shown in Figures 6 and 7, respectively, each showing graceful degradation. Figure 11 shows that both of these methods outperform Skipping the layers, suggesting that layers are still able to contribute even when run on different input sources (i.e., different layers) from how they were trained. Therefore, we answer this subsection's question as:

**Somewhat. Both randomizing and reversing the middle layer order has graceful degradation.**

Interestingly, Random Order outperforms Reverse Order as can be seen more clearly in Figure 11. One possible explanation is that Reverse the exact opposite of the order in which the layers were trained. So any random order will have at least as much consistency (in that layer $i$ is after layer $j$, where $i > j$) as totally reversing the order.

[2]Again, we emphasize that there is no fine-tuning, so the layers can't merely adapt to the new order.

### 3.5 Can We Run the Layers in Parallel?

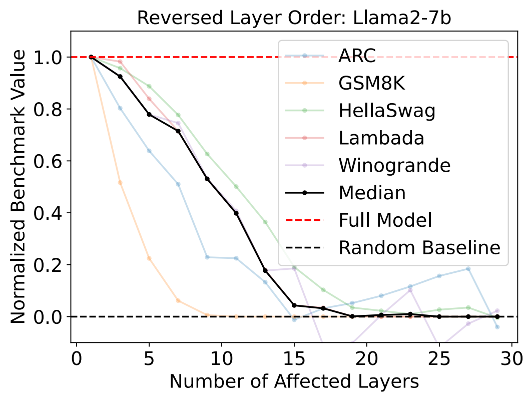
If the presence of the layers (i.e., that they're not Skipped) is more important than the order in which they're executed, we may ask whether we can run the layers *independently* from an early input and merge their results, as illustrated in Figure 1d. To answer this, we ran an experiment where, instead of skipping layers $N$ through $T - N$, we ran these middle layers in parallel, then sent their averaged result to the final $N$ layers.

Figure 8 shows graceful degradation for all benchmarks except the GSM8K math word problems. In Figure 11 this variation ("Parallel Layer") outperforms skipping layers, but curiously does worse than running the layers in reverse order. In subsection 3.6, we further explore which benchmarks are most affected by our changes, so we answer this subsection's questions with:

**Yes, except for our math-heavy benchmarks.**

### 3.6 Does the Order Matter for Some Tasks More Than Others?

Note that abstract (ARC) or mathematical (GSM8K) reasoning benchmarks have the steepest decline for most variants,



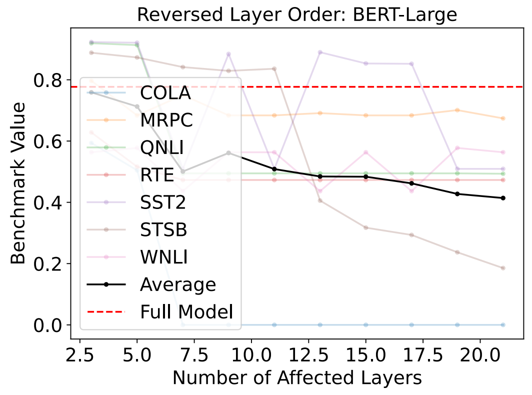
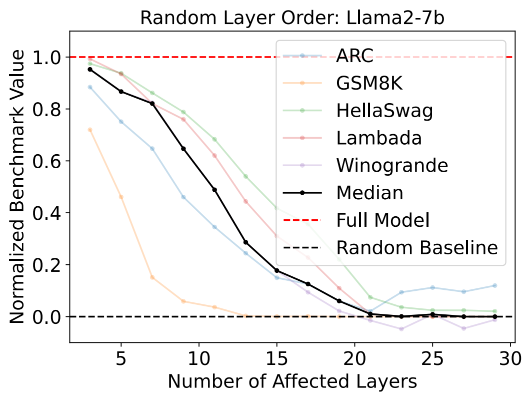


Figure 6: Top: Reversing $M$ middle layers for Llama2-7B, normalized across different Benchmarks. Bottom: Reversing layers for BERT-Large, unnormalized average.



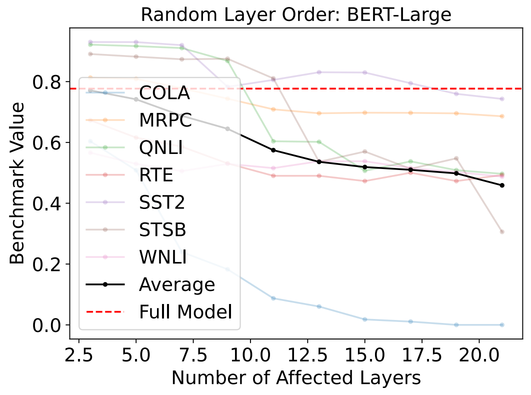


Figure 7: Randomizing layer order for $M$ middle layers for Llama2-7B (top) and BERT (bottom). Each point is the average of 10 random seeds.

including *Reversed*, *Skip*, and *Parallel*. One interpretation is that step-by-step reasoning tasks are more sensitive to layer order than "semantic" tasks like Winogrande or HellaSwag (Commonsense). This is because reasoning involves both structure and semantics to perform well compared with tasks like HellaSwag where semantics are enough to complete the task. This would be consistent with the hypothesis that some degree of order-dependent reasoning is happening within a single pass of the model. In our Painter analogy, a semantic task would be analogous to painting a collage, where ordering is less dependent, where a reasoning task might be more like painting a precise architectural scene. Regardless of whether the analogy holds, we empirically conclude that:

**Yes! Mathematical and reasoning tasks are more order dependent than "semantic" tasks.**

In Appendix A.7 we show a specific example that indicates that errors for GSM8K may come from *arithmetic* errors.

### 3.7 Does Looping Help Parallelized Layers?

Following the Painter analogy, it's conceivable that some layers only "add" to the painting when given the appropriate input. For example, the "wheel" painter will be more likely to draw some wheels if she sees the body of a car first. In transformer terms, layers might only contribute to a forward pass –as opposed to "passing" the input forward via the residual connection– when given the appropriate input. If this is the case, then iterating the parallelized layer from the previous experiment should improve performance compared to a single execution of the parallelized layer. We test this by feeding the mean output of the parallelized layer back into the same layer for a fixed number of iterations, as shown in Figure 1e.

In Figure 9, we show the results for looping the parallelized layer 3 times. As can be seen in Figure 11, this method (*Looped Parallel 3X*) significantly improves on a single iteration (*Parallel Layer*). The one exception is when the starting layer $N$ is 15 for Llama2-7B or 11 for BERT (the left-most cases for each, where only a single layer is affected). In this case, the *Looped Parallel 3X* model is equivalent to repeating only the middle layer 3 times, while the *Parallel Layer* for this point is equivalent to the full model.

We also repeated the same experiment for different numbers of iterations. In Figure **??**, we show performance for Llama2-7B as a function of the number of parallelized layers $M$ and the number of iterations. The highest performing loop iterations for each $M$ is shown by a red box. With the exception of $M = 29$ and $M = 31$ (parallelizing nearly all the layers), the optimal number of iterations is roughly linearly proportional to the number of parallelized layers. Therefore, we answer that:

**Yes, with the optimal number of iterations proportional to the number of parallelized layers.**

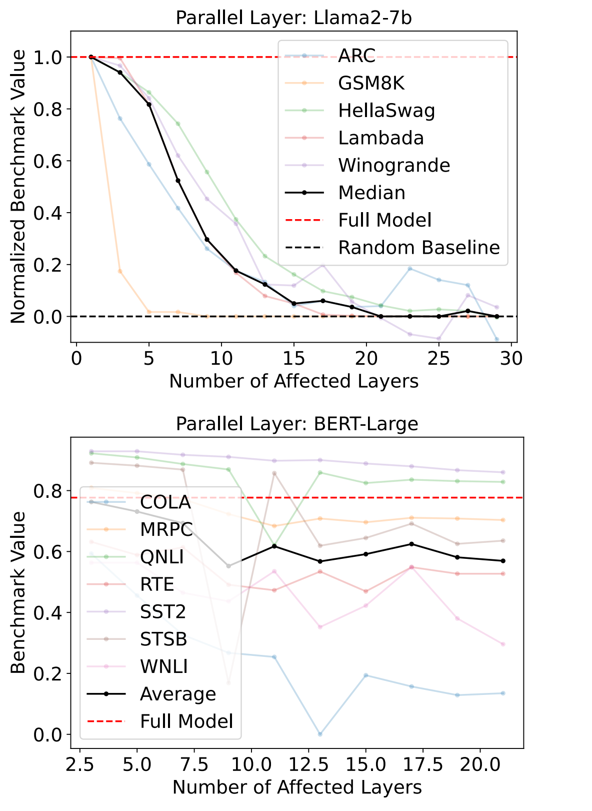


Figure 8: Running $M$ layers (Layers $(T-M)/2$ to $(T-M)/2$) in parallel for Llama2-7B (top) and BERT (bottom)

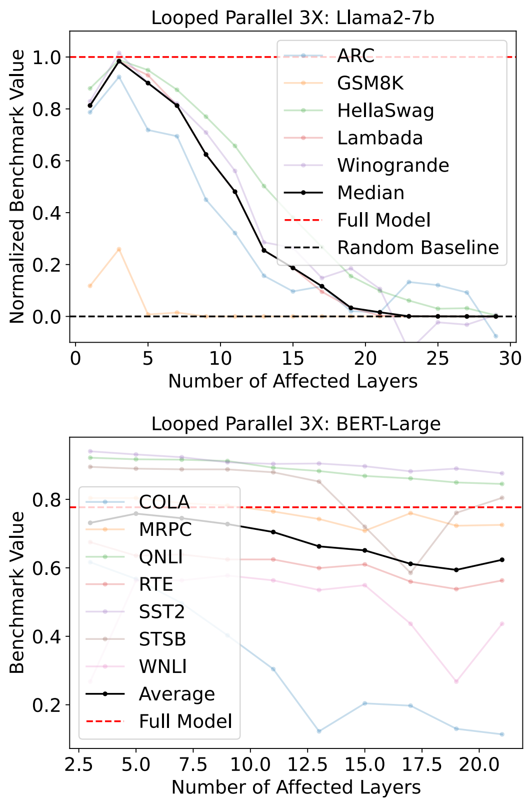


Figure 9: Running $M$ layers in parallel, looping 3 times for Llama2 (top) and BERT (bottom).

## 3.8 Which Variants Are Least Harmful?

Finally, in Figure 11 we compare all the different variants in our experiments on a single plot, showing the median (for Llama2) or average (for BERT) performance over all the benchmarks. Middle Repeat –replacing a period of middle layers with exactly the same number of copies of the middlemost layer– does worst by far, quickly degrading to random baseline performance. On the other hand, looped-parallel and random layer order have the shallowest degradation, with the former the best variant for both BERT and Llama2-7B. So we answer:

**Repeating a single layer is worst. Randomizing the layer order and looped-parallel do the least damage.**

These experiments generally show graceful degradation, but we still have the question of why the layers are somewhat robust to most of our perturbations. We offer a few suggestions in the Discussion section, but leave a full explanation for future work.

## 4 Related Work

A transformer layer contains a pair of multi-head attention (MHA) and feed-forward network (FFN), and almost all of the prior works focused on finding a combination of them that works best, or reducing the parameter count in one way or another. Our work offers an additional perspective, in that we also investigate parallelizing and reusing layers.

(Kim et al. 2024) showcased that pruning entire transformers layers can reduce latency without a considerable drop in performance. This is in line with the findings in (Bhojanapalli et al. 2021). Also, both the works noted that the performance drop is substantial if we drop the first few entire transformer layers. Hence there is an agreement that the first few transformers layers are crucial for performance. One implication of this observation is that many of these layers would be carrying redundant information, and this was shown by (Kim et al. 2024) who removed these layers, and noticed the change in the PPL score. The authors then removed these layers in one-shot, and retrained the model with LoRA to make up for the lost performance,

One aspect where (Bhojanapalli et al. 2021) and (Kim et al. 2024) observations differ though is the fine-grained units. (Bhojanapalli et al. 2021) observed that removing MLP layers have lesser impact on performance compared to removing an entire transformer layer, whereas (Kim et al. 2024) observed that this behavior is very much dependent on the size of the models. They noted that removing individual MHA and FFN modules results in better downstream task accuracy but worse PPL compared to removing entire transformer layers when the model has more than 5B parameters. For smaller models than 5B, layer-level pruning achieves superior results. While (Kim et al. 2024) did a successful job on pruning the models, the authors observed an (un)interesting side effect of the same. The pruned models perform worse when responding to

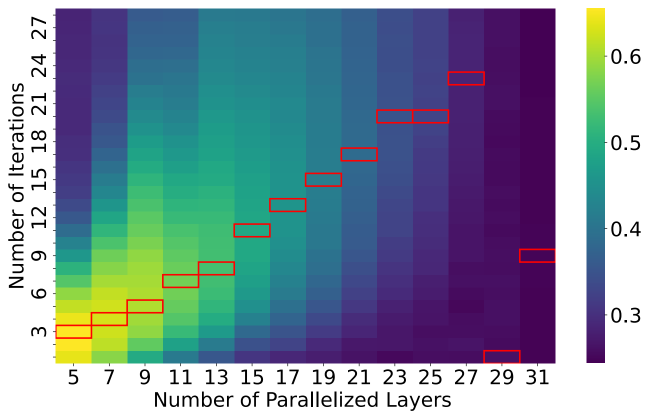


Figure 10: Looping parallelized layers of Llama2-7B, iterating from 1 to 28 times. For each number of parallelized layers, the best iteration number is marked by a red box.

factual questions or generating long responses. The authors couldn't make up for the lost performance on these tasks even after retraining the models, suggesting that while much of the information stored in these layers was redundant, some parts of it were required for critical tasks e.g. factual Q&A.

The experiments of ShortGPT (Men et al. 2024) corroborate the findings of ShortenedLlama, exploiting the redundancy in LLMs to derive a pruning technique. Denseformer (Pagliardini et al. 2024) had similar findings where they found that modules even after applying DWA had cosine similarity with original transformer modules, suggesting both that there is some redundant information flow, and that this can be leveraged for sparsity.

More recently, (Freiberger et al. 2024) explores layer shuffling during training to enhance robustness of the models, while (Dutta, Gupta, and Agarwal 2024) proposes an algorithm that can be used for efficient pruning of transformers. (Lad, Gurnee, and Tegmark 2024) explores the robustness of transformer-based LLMs by deleting or swapping layers. (Zou et al. 2024) focuses on efficient inference by splitting layers in groups, running them in parallel or bypassing them. On a similar note, (Flynn et al. 2024) focuses on pruning transformers in different ways (entire attention blocks, ffn, etc.). Our work is more closely related to (Lad, Gurnee, and Tegmark 2024) and (Flynn et al. 2024) where the ablations involve frozen models. We present a super set of such ablations for the frozen transformer models.

## 5 Discussion

In this paper, we examined several questions raised by the Layers as Painters analogy. Among our more interesting findings are: 1. There are three distinct classes of layers (with Middle being the largest). 2. The middle layers have some degree of uniformity (but not redundancy). And 3. Execution order matters more for math and reasoning tasks than semantic tasks. We welcome future theoretical analysis of layer behaviors in transformer architectures based on our empirical findings.

We leave a full explanation for why transformers are robust to our variations for future work. One possible hypothesis is that the residual connections during training are necessary for the layers to share a common representation. It's already known that residual connections are useful to help address the vanishing gradient problem (He et al. 2015), and that transformers trained without these connections perform worse than without. However, it would be interesting to rerun our variations on models without residuals, and see if our variations destroyed whatever meager gains full non-residual models achieved.

We also plan to "thaw" our models and investigate if transformers take to adjust to the variations in the paper via fine-tuning. If these models were fine-tuned with new architectures, the performance would probably be even better. It is worth noting that Parallel and Skip both have potentially lower latencies than the full model (assuming enough memory to execute the layers simultaneously). For example, the latency for the Parallel Layer for Llama2-7B for N=8 should be about half that of normal Llama2-7B. Though the aim of this paper is to better understand layers in transformer-based LLMs as opposed to introducing new models, our results suggest simple methods to easily trade accuracy for latency gains. Our results also suggest that a routing mechanism for executing frozen layers may be used here, analogous to Switch Transformers (Fedus, Zoph, and Shazeer 2022).

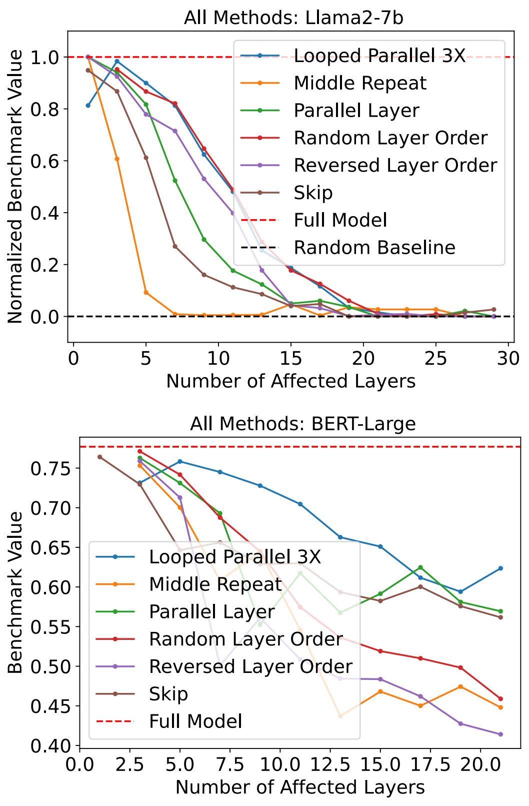


Figure 11: Average benchmark scores for different variations for Llama2-7B (top) and BERT-large (bottom).

## Acknowledgements

We would like to thank Owen He, who came up with the painter analogy after seeing some of our early results. We would also like to thank Yujin Tang for providing valuable suggestions during the rebuttal process.

## References

Akiba, T.; Shing, M.; Tang, Y.; Sun, Q.; and Ha, D. 2024. Evolutionary Optimization of Model Merging Recipes. arXiv:2403.13187.

Bhojanapalli, S.; Chakrabarti, A.; Glasner, D.; Li, D.; Unterthiner, T.; and Veit, A. 2021. Understanding Robustness of Transformers for Image Classification. arXiv:2103.14586.

Biderman, S.; Schoelkopf, H.; Anthony, Q.; Bradley, H.; O'Brien, K.; Hallahan, E.; Khan, M. A.; Purohit, S.; Prashanth, U. S.; Raff, E.; Skowron, A.; Sutawika, L.; and van der Wal, O. 2023a. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling. arXiv:2304.01373.

Biderman, S.; Schoelkopf, H.; Anthony, Q. G.; Bradley, H.; O'Brien, K.; Hallahan, E.; Khan, M. A.; Purohit, S.; Prashanth, U. S.; Raff, E.; et al. 2023b. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, 2397–2430. PMLR.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *ArXiv*, abs/1803.05457.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Dutta, O.; Gupta, R.; and Agarwal, S. 2024. VTrans: Accelerating Transformer Compression with Variational Information Bottleneck based Pruning. arXiv:2406.05276.

Fedus, W.; Zoph, B.; and Shazeer, N. 2022. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. arXiv:2101.03961.

Flynn, M.; Wang, A.; Alvarez, D. E.; Sa, C. D.; and Damle, A. 2024. STAT: Shrinking Transformers After Training. arXiv:2406.00061.

Freiberger, M.; Kun, P.; Løvlie, A. S.; and Risi, S. 2024. LayerShuffle: Enhancing Robustness in Vision Transformers by Randomizing Layer Execution Order. arXiv:2407.04513.

Friedman, D.; Lampinen, A. K.; Dixon, L.; Chen, D.; and Ghandeharioun, A. 2023. Comparing Representational and Functional Similarity in Small Transformer Language Models. In *UniReps: the First Workshop on Unifying Representations in Neural Models*.

Godey, N.; Éric de la Clergerie; and Sagot, B. 2024. Anisotropy Is Inherent to Self-Attention in Transformers. arXiv:2401.12143.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. arXiv:2310.06825.

Kim, B.-K.; Kim, G.; Kim, T.-H.; Castells, T.; Choi, S.; Shin, J.; and Song, H.-K. 2024. Shortened LLaMA: A Simple Depth Pruning for Large Language Models. arXiv:2402.02834.

Kornblith, S.; Norouzi, M.; Lee, H.; and Hinton, G. 2019. Similarity of Neural Network Representations Revisited. arXiv:1905.00414.

Lad, V.; Gurnee, W.; and Tegmark, M. 2024. The Remarkable Robustness of LLMs: Stages of Inference? arXiv:2406.19384.

Men, X.; Xu, M.; Zhang, Q.; Wang, B.; Lin, H.; Lu, Y.; Han, X.; and Chen, W. 2024. ShortGPT: Layers in Large Language Models are More Redundant Than You Expect. arXiv:2403.03853.

Pagliardini, M.; Mohtashami, A.; Fleuret, F.; and Jaggi, M. 2024. DenseFormer: Enhancing Information Flow in Transformers via Depth Weighted Averaging. arXiv:2402.02622.

Paperno, D.; Kruszewski, G.; Lazaridou, A.; Pham, Q. N.; Bernardi, R.; Pezzelle, S.; Baroni, M.; Boleda, G.; and Fernández, R. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. arXiv:1606.06031.

Sakaguchi, K.; Bras, R. L.; Bhagavatula, C.; and Choi, Y. 2019. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *arXiv preprint arXiv:1907.10641*.

Simoulin, A.; and Crabbé, B. 2021. How Many Layers and Why? An Analysis of the Model Depth in Transformers. In Kabbara, J.; Lin, H.; Paullada, A.; and Vamvas, J., eds., *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, 221–228. Online: Association for Computational Linguistics.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Ferrer, C. C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.-A.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 353–355. Brussels, Belgium: Association for Computational Linguistics.

Xue, F.; Chen, J.; Sun, A.; Ren, X.; Zheng, Z.; He, X.; Chen, Y.; Jiang, X.; and You, Y. 2023. A Study on Transformer Configuration and Training Objective. arXiv:2205.10505.

Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; Mihaylov, T.; Ott, M.; Shleifer, S.; Shuster, K.; Simig, D.; Koura, P. S.; Sridhar, A.; Wang, T.; and Zettlemoyer, L. 2022. OPT: Open Pre-trained Transformer Language Models. arXiv:2205.01068.

Zou, L.; Wang, Q.; Zhao, H.; Kong, J.; Yang, Y.; and Deng, Y. 2024. CQIL: Inference Latency Optimization with Concurrent Computation of Quasi-Independent Layers. arXiv:2404.06709.