The Remarkable Robustness of LLMs: Stages of Inference?

Vedang Lad [1] **Wes Gurnee** [1] **Max Tegmark** [1 2]

Abstract

We demonstrate and investigate the remarkable robustness of Large Language Models by deleting and swapping adjacent layers. We find that deleting and swapping interventions retain 72-95% of the original model's prediction accuracy without fine-tuning, whereas models with more layers exhibit more robustness. Based on the results of the layer-wise intervention and further experiments, we hypothesize the existence of four universal stages of inference across eight different models: detokenization, feature engineering, prediction ensembling, and residual sharpening. The first stage integrates local information, lifting raw token representations into higher-level contextual representations. Next is the iterative refinement of task and entity-specific features. Then, the second half of the model begins with a phase transition, where hidden representations align more with the vocabulary space due to specialized model components. Finally, the last layer sharpens the following token distribution by eliminating obsolete features that add noise to the prediction.

1. Introduction

Advancements in Large Language Models (LLMs) have demonstrated remarkable reasoning capabilities, often attributed to their increased scale (66). However, this also heightens risks and vulnerabilities (7; 36; 4), necessitating extensive research into the underlying mechanisms of these capabilities. Inspired by studies on model robustness (28; 45; 57; 44; 5; 64), this work investigates the sensitivity of LLMs to the deletion and swapping of entire layers during inference. Our findings suggest four universal stages of inference: detokenization, feature engineering, prediction ensembling, and residual sharpening.

Recent work in mechanistic interpretability has explored the iterative inference hypothesis (3; 57), which suggests that each layer incrementally updates the hidden state of a token towards decreasing loss by gradually shaping the next token distribution (24). Self-repair (57) and redundancy (45; 28) in networks further support this hypothesis of iterative inference. However, recent work also indicates a degree of specialization in networks, with attention heads and neurons playing specific roles (30; 43; 26), which compose into more sophisticated circuits (52; 20). In this work, we begin by exploring the robustness of language models by performing a series of interventions that delete individual layers or swap adjacent layers (Figure 2). Using these results, we then attempt to understand the roles of different depths in the network. Our experiments suggest four distinct phases in a model, which we investigate further.

Specifically, we hypothesize an initial **(1) detokenization** (15) stage, where the model integrates local context to convert raw token representations into coherent entities, as suggested by the sensitivity to deletion and swapping. In the **(2) feature engineering**, the model iteratively builds feature representations based on token context, leading to minimal progress in token prediction but significant increases in probing accuracy and patching importance. A phase transition from attention-heavy computation to MLP-heavy computation delineates the following stage. During the **(3) prediction ensembling**, the model emphasizes relevant predictions while suppressing others, potentially marked by high MLP computation and the emergence of prediction neurons. Finally, in the **(4) residual sharpening** stage, tokens transition from semantic representations to specific next-token predictions, again showing sensitivity to deletion and swapping. This framework represents a tentative step toward understanding the complex nature of token processing in advanced language models. Figure 1 delineates our four characteristic phases, which we describe further in Table 1.

2. Related Work

Universal Mechanisms A key activity of mechanistic interpretability is circuit analysis, where research uncovers relevant model components for a given computation. In computer vision, circuits discover how features are constructed across many layers (51). Follow-up work found that feature building was carried out by specific mechanisms

[1]MIT [2]Institute for Artificial Intelligence and Fundamental Interactions (IAIFI). Correspondence to: Vedang Lad <vedang@mit.edu>.

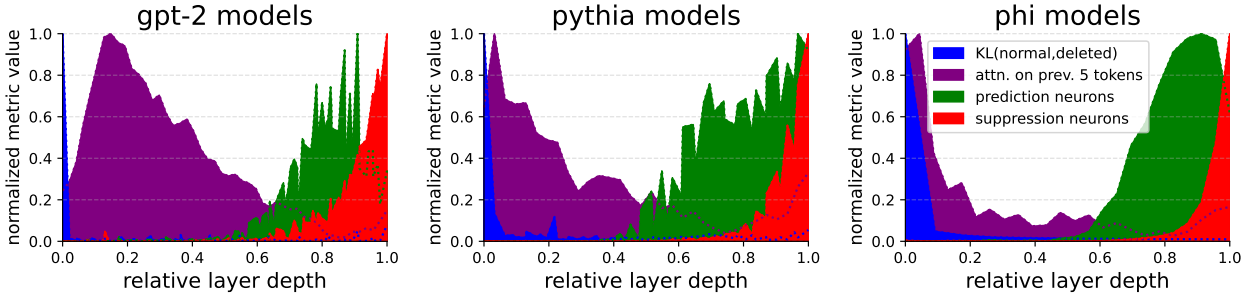


Figure 1. Performing layer-wise interventions such as deleting and swapping layers hints at four stages of inference. (Blue) KL between normal model and layer ℓ zero-ablated. (Purple) Total attention paid to the previous five tokens in a sequence. (Green) The number of "prediction" neurons (Red) The number of suppression neurons (25; 65; 30).

Table 1. Our Hypothesis: Universal Inference Stages

Stage	Name	Function	Observable signatures
1	Detokenization	Integrate local context to transform raw token representations into coherent entities	Catastrophic sensitivity to deletion and swapping
2	Feature Engineering	Iteratively build feature representation depending on token context	Little progress made towards next token prediction, but significant increase in probing accuracy and patching importance. Attention Heavy Computation
3	Prediction Ensembling	Convert previously constructed semantic features into plausible next token predictions using an ensemble of model components.	Increased MLP importance; prediction neurons appear; phase transition in progress towards final prediction
4	Residual Sharpening	Sharpen the next token distribution by eliminating obsolete features that add noise to the prediction	More suppression neurons than prediction neurons

that appeared across models, such as frequency detectors (59) and curve-circuits (8). Language models seem to be following a similar line of inquiry, first uncovering universal model components, such as induction heads (52), successor heads (26), and copy suppression (43) in attention mechanisms. The discovery of knowledge neurons (10) paved the way for the identification of various specialized neurons (30; 65). These specialized components can be connected to critical roles in universal processes in language models, such as circuit reuse (47), variable finding mechanisms (19), self-repair (57; 44) (which also studies layer-wise ablations), function vectors (62; 35), and long context retrieval (63).

Depth-Dependent Studies Circuit analysis naturally motivated various depth-dependent studies, such as the logit lens (50), a technique that reveals the model's prediction distribution at each layer. This line of reasoning revived the iterative inference hypothesis, the idea that each layer updates the hidden state in a direction of decreasing loss, in the context of ResNets (27; 37). Due to residual connections, researchers were able to demonstrate that models exhibited ensembling" (64) through layer ablations and permutations, referred to as lesion" studies. This was later applied to modern transformers (13; 5). Recent work (3) expanded on the logit lens and provided further evidence for iterative inference. This hypothesis was also supported by analyzing transformers in embedding spaces (12) and model self-repair (57).

Many works have localized specific kinds of computations to particular regions of large language models. For instance, model editing research (46) showed that knowledge is stored in mid-layer MLP neurons. Follow-up studies (33; 23) suggested that these facts can be stored across layers, with different components encoding task-specific instructions and recall. There is significant literature citing changes halfway through a language model, such as linear probe quality improving the fastest in the first half of a model (32) or fine-tuning predominantly updating weights in the middle of a model (53). Few works observed that activation sparsity changes in the middle of a model, transitioning from sparse to dense (40; 65).

Other works have extended these insights into phases, such as identifying emergent specific phases of truth processing in language models (41) and distilling translation in multilingual transformers into three distinct phases: input space, concept space, and output space (67). Other works hypothesized stages of inference in large language models (15). This was further studied in the context of iterative inference (3), which demonstrated the importance of the first layers through layer ablations. When permuting the layers of GPT-2 style transformers, studies found the best performance in variants that have a higher proportion of self-attention layers at the beginning of models and more feedforward layers at the end (54). These studies thematically suggest a preferential order to computation, which we investigate further.

Robustness: Pruning and Redundancy Numerous studies have inadvertently revealed depth-related findings while exploring model pruning. Prior work on zero-ablations of transformers has predominantly concentrated on BERT (14) style transformers (70; 17; 18; 58; 68). Despite significant differences in these models (16), many underlying principles may be applicable. For example, pruning the final layers of a model retains most of the model's performance (58). Follow-up studies suggest 85% neuron redundancy in BERT transformers (11). Analogously, recent findings demonstrated that approximately 70% of attention heads and 20% of feed-forward networks can be removed with minimal impact on task performance (1), suggesting redundancy (28; 45). Model pruning work found improved benchmark performance by only keeping low-rank components of MLPs in the second half of the models (60) while uncovering token frequency dependencies in MLP weights. While we only touch upon the relationship between robustness and token frequency, (42) proposes that pretraining data contributes to this robustness.

3. Experimental Protocol

Models To investigate the stages of inference in language models, we examine the Pythia (6), GPT-2 (56), and Microsoft Phi (29; 39) model families, which range from 124M to 6.9B parameters (see Table 2). All families use decoder-only transformers but exhibit several architectural differences that enable us to test the generality of our findings. Pythia models utilize parallel attention and MLPs, executing these components concurrently during inference. In contrast, GPT-2 and Phi models apply these components sequentially, with attention followed by the MLP (see Figure 2). Additionally, GPT-2 models were trained with dropout. Despite this, all models exhibit consistent inference patterns that support our hypothesis. We preprocess weights identically across all models, as described further in Appendix D.

Data We evaluate all three model families on a corpus of one million tokens from random sequences of the Pile dataset (22). The Pile was used to train the Pythia models and includes OpenWebText, the training corpus for GPT-2. Testing models on data similar to their training data ensures a fair comparison and minimizes the impact of domain shifts on observed inference patterns.

Layer Swap Data Collection To study the robustness and role of different model components at different depths, we employ a swapping intervention where we switch the execution order of a pair of adjacent layers in the model. Specifically, for a swap intervention at layer ℓ, we execute the transformer block (including the attention layer, MLP, and normalization) $\ell + 1$ before executing block ℓ. We record the Kullback-Leibler (KL) divergence between the intervened and original models, measuring the difference in their output distributions, along with model-wise metrics such as loss, top-1 prediction accuracy, and prediction entropy. This intervention allows us to examine how the order of computation affects the model's behavior and performance at different depths.

Ablation Data Collection To generate baselines for each layer swap experiment, we perform zero ablations on the corresponding layer while collecting the same metrics. The ablation preserves the swap ordering: for a swap ordering of **1-2-4**-3-5, the ablation maintains **1-2-4**-5. We opt for zero ablation as opposed to mean ablation, as proposed by (3), to maintain consistency with the swap order. Additionally, we perform attention-only and MLP-only ablations to study the specific roles of these components in the model's inference process. By comparing the effects of layer swapping and ablation, we can gain insights into the importance and function of each component at different depths in the model.

4. Robustness

4.1. Intervention Results

To study the robustness of language models, we apply our aforementioned drop and swap interventions to every layer of four GPT2 models (55) and four Pythia (6). In Figure 3, we report (1) the KL divergence between the prediction of the intervened model and the nominal model, (2) the fraction of predictions that are the same between the intervened model and the baseline model (denoted as relative accuracy), and (3) the change in entropy of the prediction between the intervened and baseline model for all interventions.

Our results show that intervening on the first layer is catastrophic for model performance. Specifically, dropping or swapping the first layer causes the model to have very high entropy predictions as opposed to causing a mode collapse on a constant token. In Pythia models, swapping the last

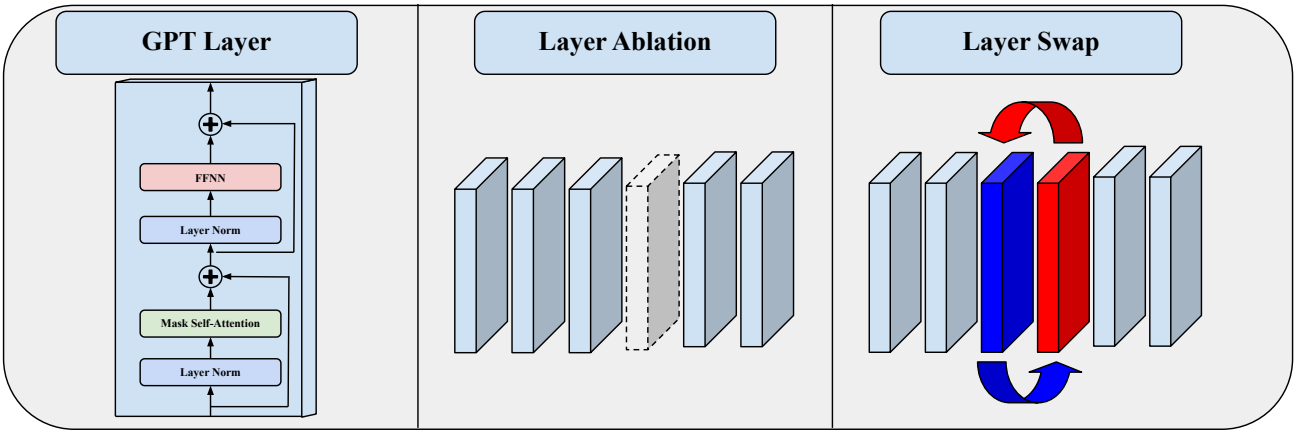


Figure 2. To study the stages of inference, we perform two experiments, each a layer-wise intervention, where a layer (left) encompasses all model components. The first intervention is a zero ablation of the layer (middle), in which a layer is fully removed and residual connections skip the layer entirely. The second intervention (last) is an adjacent layer swap, in which we permute the positions of two layers. The ablation is performed on all layers, while the layer swap is performed on all adjacent pairs of layers in the model.

Table 2. Comparison of Model Series

Table 2. Pythia Model Series

Parameters	Layers
Pythia (410M)	24
Pythia (1.4B)	24
Pythia (2.8B)	32
Pythia (6.9B)	32

Table 2. GPT-2 Model Series

Parameters	Layers
Small (124M)	12
Medium (355M)	24
Large (774M)	36
XL (1.5B)	48

Table 2. Microsoft Phi Model Series

Parameters	Layers
Phi 1 (1.3B)	24
Phi 1.5 (1.3B)	24
Phi 2 (2.7B)	32

layer with the second to last layer also has a similar catastrophic high-entropy effect, while GPT2 models largely preserve their predictions. We further discuss our hypothesized role of the first and last layer in Section 5.1 and Section 5.4 respectively.

In contrast to the first and last layer interventions, the middle layers are remarkably robust to both deletion and minor order changes. When zooming in on the differences between the effect of swaps and drops for intermediate layers, we find that swapping adjacent layers is less harmful than ablating layers. These results match similar experiments performed on vision transformers (5). We take this as evidence that certain operations within the forward pass are commutative, though further experimentation is required.

We suspect that GPT2 exhibits greater robustness than Pythia because (1) GPT2 models are trained with dropout, which likely increases redundancy and (2) GPT2 models have fewer parameters per layer so a GPT2 layer ablation removes fewer parameters than a Pythia ablation.

4.2. Why are Language Models Robust to Layer-Wise Interventions?

We suspect that the robustness of language models can be partially attributed to the presence of residual connections (64) in the transformer architecture, which lead to increased redundancy (28; 45). While skip connections were initially introduced to mitigate the vanishing gradient problem, investigations of deep residual networks (ResNets) found that residual connections facilitate "ensembling" within networks (34; 64). We hypothesize that, as is the case for ResNets, residual connections in language models allow gradient descent to form ensembles of relatively shallow computational sub-networks. In doing so, networks avoid strong dependencies on individual paths, thereby increasing their resilience to layer-wise interventions. This hypothesis is supported by recent observations of self-repair mechanisms (44; 57) that demonstrate the existence of parallel computational paths.

5. Stages of Inference Hypothesis

We now discuss and provide tentative evidence for our hypothesis of four universal stages of inference in transformer language models. Two caveats apply to all of the following

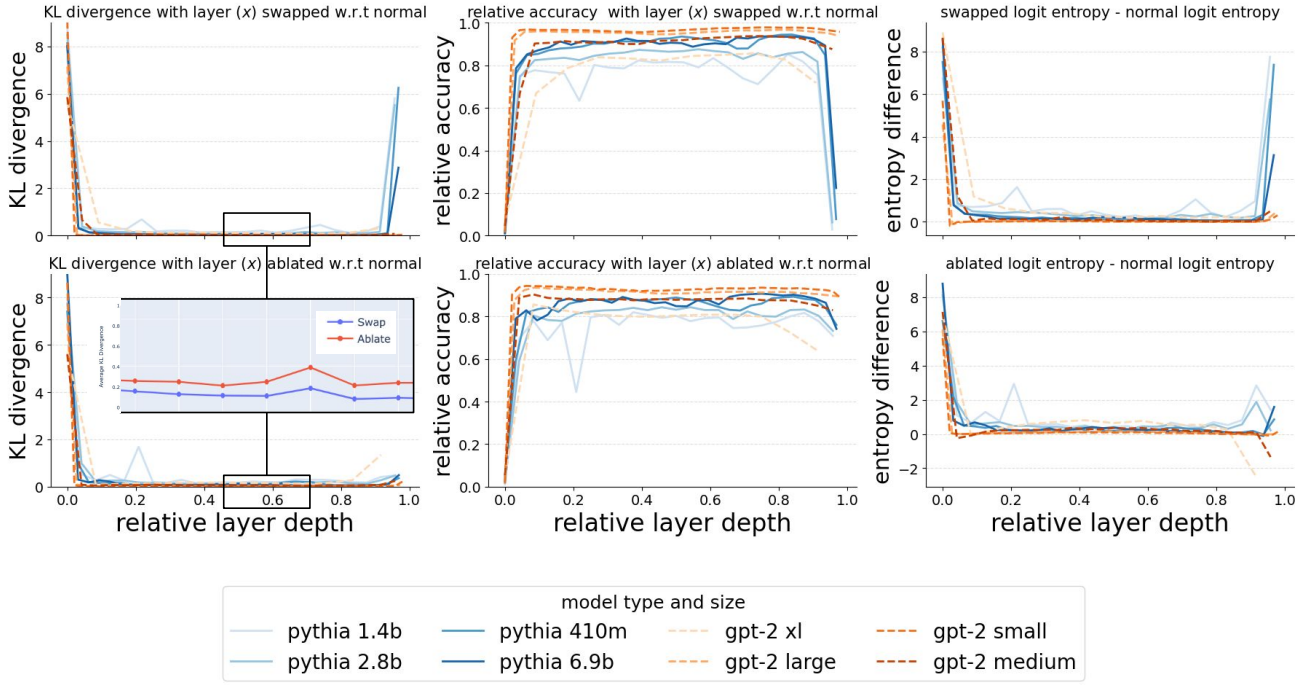


Figure 3. Effect of layer swap (top) and layer drop (bottom) interventions on KL divergence (left), consistency of the top-1 prediction (middle), and the change in entropy (right) between the intervened and baseline model. (zoom) all models (Pythia 1.4b shown) have layer swaps resulting in lower KL than ablation.

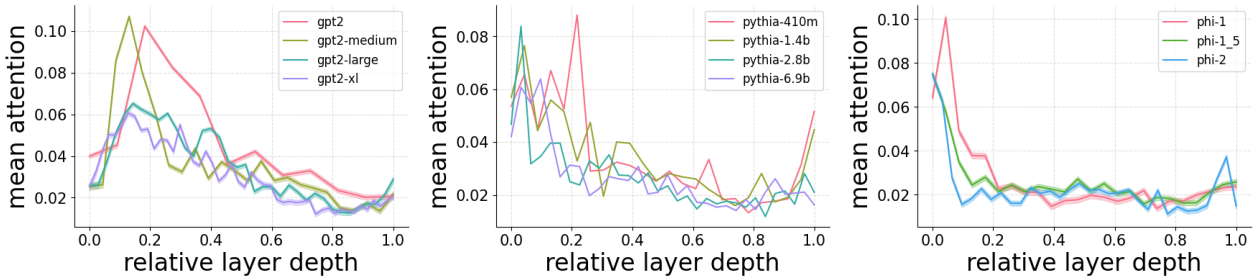


Figure 4. The mean attention of the previous five tokens in a sequence, as a function of relative depth of layers.

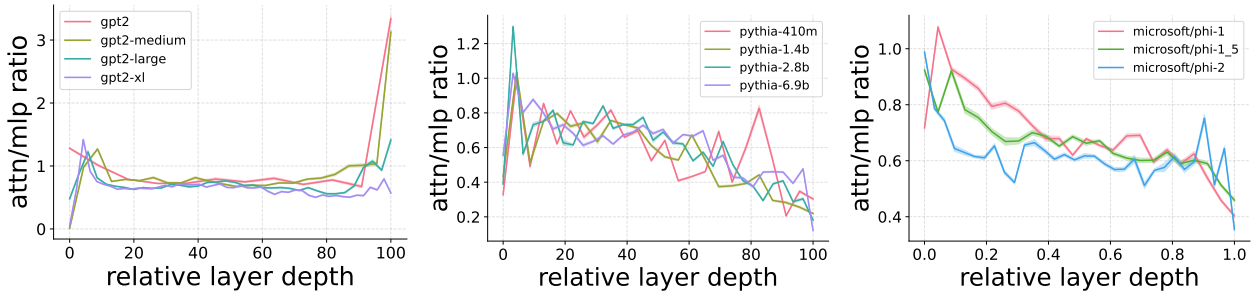


Figure 5. The ratio of the output norm of attention heads over the MLP, as a function of the relative depth of layers. Models present high attention function in early stages, and less in later stages. GPT models see an increase in the final layer, which we hypothesize the cause of in Section 7.

subsections: First, the boundaries between stages are fuzzy, and in practice, more than one stage can occur simultaneously. Second, these stages represent aggregate computational patterns, but the processing of any specific kind of token is likely to undergo more individualized dynamics (e.g., factual recall (46; 49)).

5.1. Stage 1: Detokenization

Given the extreme sensitivity of the model to first-layer ablations, we infer that the first layer is not so much a normal layer as it is an extension of the embedding. This is especially true for the Pythia model family due to the use of parallel attention, which implies that the first MLP layer is *only* a function of the current token. Consequently, by ablating the first layer, the rest of the network is blind to the immediate context and is thrown off distribution.

Immediately after computing this extended embedding, evidence from the literature suggests that the model concatenates nearby tokens that are part of the same underlying word (9; 21) or entity (49) (e.g., a first and last name). This operation integrates local context to transform raw token representations into coherent entities. In this way, the input is "detokenized" (15; 31).

Previous work has shown the existence of neurons that activate for specific n-grams (31; 65). Of course, to accomplish this, there must be attention heads that copy nearby previous tokens into the current token's residual stream. More generally, if early layers are integrating local context, we would predict that early-layer attention heads pay disproportionately more attention to nearby tokens than later attention heads. To test this, we compute the fraction of attention paid to tokens within the previous five positions of the present token. As can be seen in Figure 5, attention is indeed more local in the early layers. Moreover, the output norm of the attention ratio is higher than the output norm of the MLP layers in the first few layers (54).

5.2. Stage 2: Feature Engineering

Building upon the locally contextualized representations from stage 1, we hypothesize that a second region of the model performs "feature engineering" to construct features that could be useful for making downstream predictions, either for the next token or for future tokens in the context. While the kinds of features will vary by token type, there are many results in the literature that localize intermediate feature construction to the early to middle layers via patching (69) and probing (2) experiments.

For example, the model editing literature suggests that the MLPs in this region are important for factual recall (46; 23; 49). This region is where probing accuracy for spatial and temporal features drastically but smoothly improves (32). The features produced in this stage influence downstream predictions, as evidenced by steering and patching experiments on sentiment (61), truth (41), and zero-shot function execution (62) that peak in effectiveness in stage two layers. The representations and features formed in this stage are increasingly abstract, transitioning from shallower syntactic features to richer semantic features (15; 67; 38).

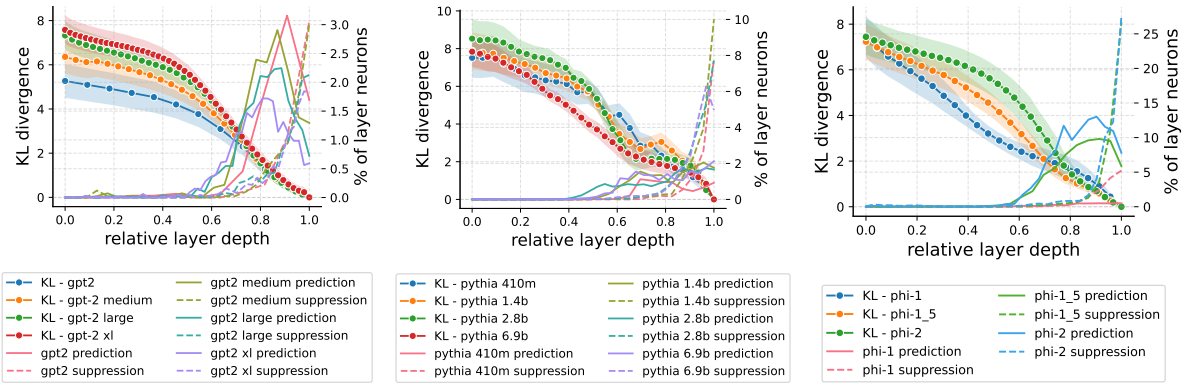
Logit Lens However, in this stage, the features are simply produced, rather than consumed to make a concrete prediction. To show this, we perform a logit lens experiment (50; 12) where we apply the unembedding to the residual stream after every layer to estimate a model's intermediate prediction. We then compute the entropy of this intermediate prediction and the KL divergence with the final prediction. As can be seen in Figures 10 and 6, there is very little progress towards making an actual prediction. For that, we require the next two stages.

5.3. Stage 3: Prediction Ensembling

After about the halfway point, the model must begin converting semantic features into concrete predictions for the next token. Given the robustness observed in Figure 3, we hypothesize that this is accomplished with a kind of ensembling. Ensembling in neural networks with residual connections is akin to having many subnetworks perform a "vote" for the output (64). Therefore, interfering with a single member of an ensemble is unlikely to have a destructive effect.

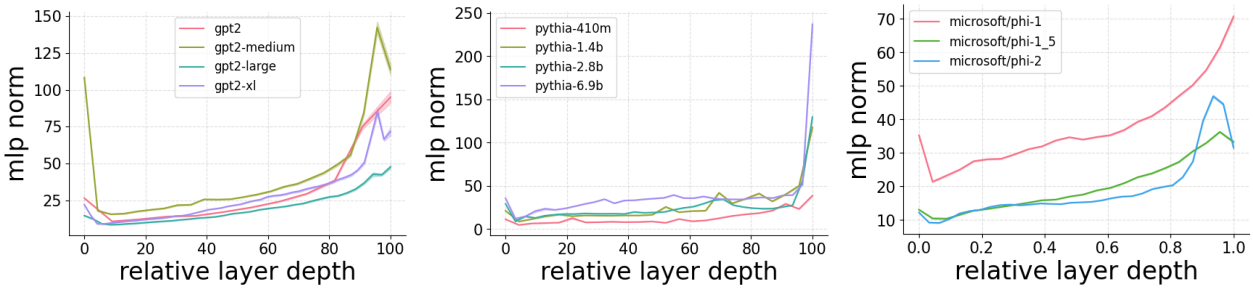
Prediction Neurons Previous work suggests that networks contain ensembles of "prediction" neurons, which act as probability promoters (65; 24; 30) and work in tandem with suppression neurons (Section 5.4). Following (30), we find prediction and suppression neurons by analyzing the output weights with the unembedding matrix $\mathbf{W}_U$. Prediction neurons exhibit a logit effect distribution $\mathbf{W}_U \cdot \mathbf{w}_{out}$ with high kurtosis and positive skew, while suppression neurons show high kurtosis but negative skew. Across 11 models, prediction neurons emerge around the midpoint, increasing in density towards the latter layers 6, before being outstripped by suppression neurons. To confirm their action, we study how much the model's abstract representation changes as a function of prediction neuron density.

Change in Intermediate Prediction To quantify how far the model's representation is from the next prediction, we plot the "distance" (KL divergence) remaining for the model to reach its output distribution in Figure 6. We find that the rise in prediction neurons corresponds to a phase transition in the decrease of KL divergence. The number of prediction neurons peaks at around 85% through the model's layers and decreases in the last few layers.



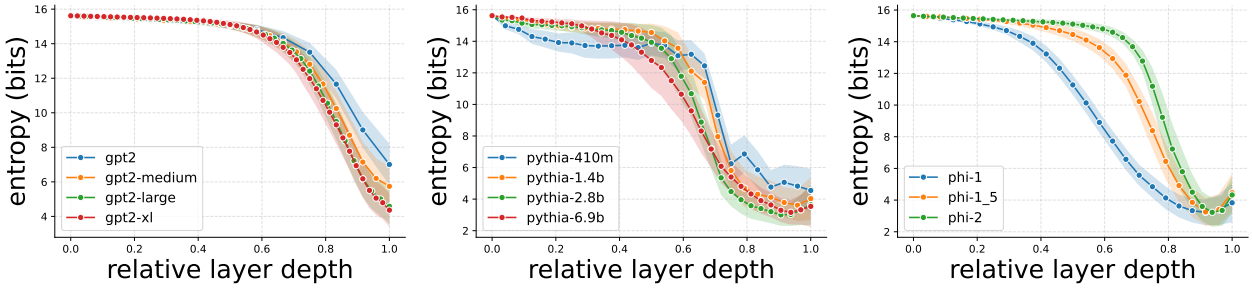
((a)) GPT models ((b)) Pythia models ((c)) Phi models

Figure 6. We measure KL divergence between intermediate and final predictions using the logit lens method (50). On the second axis, we use an automated procedure for classifying neuron types detailed in (30), into prediction neurons and suppression neurons. These are universal neurons in all models known to increase the probabilities of tokens and decrease the probabilities of others. We hypothesize this inverse relationship as evidence for ensembling in networks.(65)



((a)) GPT MLP Output ((b)) Pythia MLP Output ((c)) Phi MLP Output

Figure 7. The norm of the output of every MLP across its layers to measure its contribution to the residual stream. Across all 11 models, the norm grows and peaks in the final layers before output, suggestive of the final two stages of inference, predictive ensembling, and residual sharpening



((a)) GPT Logit Lens Entropy ((b)) Pythia Logit Lens Entropy ((c)) Phi Logit Lens Entropy

Figure 8. Using the logit lens technique (50), we calculate the probability distribution of the next token at the end of every layer, and then take its entropy. This provides a measure of the model's confidence in the next prediction, which coincides with the rise in suppression neurons, a large MLP output norm which are characteristic of residual sharpening.

Phi-1 contains fewer prediction neurons than other Phi models and also has a lower slope in its KL divergence 6(c). GPT 6(a) and Phi 6(c) models exhibit more prediction neurons and steeper, smoother KL divergence slopes compared to Pythia 6(b). Surprisingly, Microsoft Phi models, which are known to outperform models with similar parameters, exhibit nearly 15% of prediction neurons per layer and 25% suppression neurons. This is 5-8x the density in GPT-2 and 3-7x the density in Pythia models, respectively. At around 90% through the model layers, however, the prediction neu-

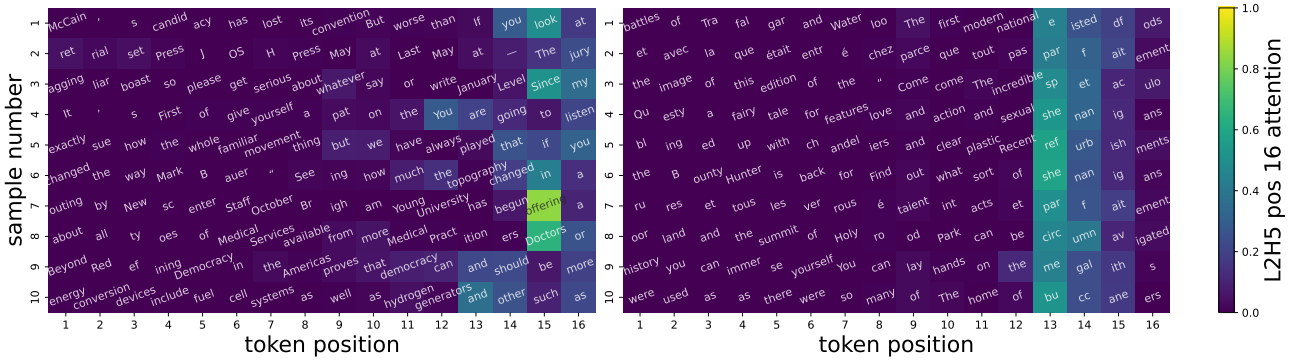


Figure 9. Attention from source token to the final token in various inputs. An identified sub-joiner attention head found in the early layers of language models is responsible for attending to multi-token words (right)

ron density decreases, while models continue approaching their final distribution, sometimes even *accelerating* 6(b). This suggests the action of other mechanisms, which we speculate is the final stage of inference.

5.4. Stage 4: Residual Sharpening

The subsiding of prediction neurons in the previous stage possibly suggests a new member of the ensemble, providing the final "push" to predicting the next token distribution. Our investigation reveals that the final layers of all models contain the highest density of suppression neurons, which may work to delete previously constructed features, suppress probabilities of invalid tokens, and/or calibrate the confidence in the final prediction.

Ensemble Bias Prediction and suppression neurons both manipulate the residual stream and, as inverses of one another, can effectively perform each other's functions. These neurons appear in different ratios and varying densities across the model. To study how these neurons sharpen the representation, we plot the logit lens entropy of the model. In certain models, such as Pythia (Figures 8(b) and 8(c)), the entropy sometimes *increases* in the final layers, suggesting overconfident predictions are blunted. In other words, the suppression neurons can either suppress tokens or features outside of the top-one to sharpen the distribution, or suppress its confidence in the top token to flatten out the prediction distribution. This finding supports previous work, which suggests that models can shift away from the correct token to an incorrect token in the final layers (50; 60), and pruning or rank-reducing these layers can, in turn, improve performance (45; 28).

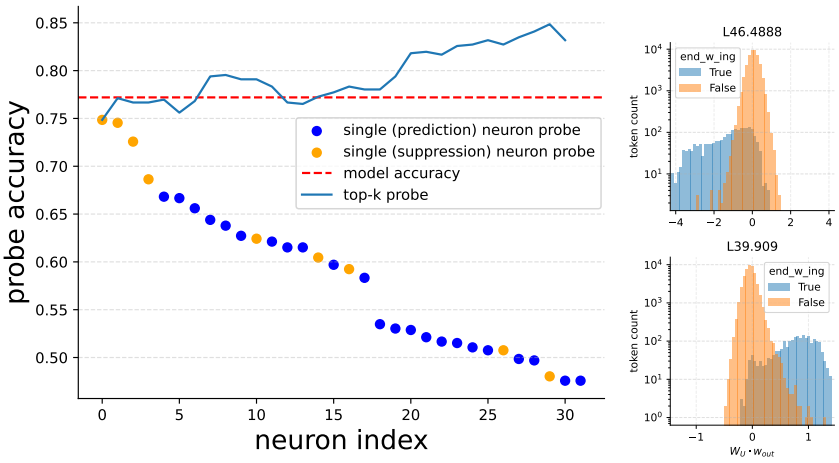
Final Layer The intensity of suppression neurons, as seen in Figure 6, is localized in the final few layers of the model, where the quantity of suppression neurons outstrips predictive neurons. To quantify the *intensity* of this change, we measure the norm of the MLP output, where a larger norm suggests a greater contribution to the residual (Figure 7). Removal of the final layer or permuting its position results in the breakage of the model (Figure 3), analogous to the breakage observed in the first layers, during which the attention norm is the greatest (Figure 5). As a result, we speculate the importance of ordering in the first and last layers due to the magnitude of change they impart.

6. Case Studies

To integrate the stages of inference hypothesis with mechanistic descriptions of models, we present two case studies. First, we identify attention heads responsible for constructing multi-token words, known as *subjoiner* heads (21). These heads help capture the context of a token for appropriate prediction, thus contributing to the detokenization and feature engineering stages of models. In the second case study, we provide evidence of the ensembling of prediction and suppression neurons. Through probing experiments, we demonstrate that multiple prediction and suppression neurons working jointly significantly outperform probes trained on individual neurons and sometimes even surpass the model's performance.

6.1. Study 1: Attention on Four-Token Words

A crucial aspect of detokenization and feature engineering is building representations that integrate the context preceding a token. For a language model to understand and predict multi-token words, it must capture both the sentence context and the tokens that comprise a single word. To study these mechanisms, we construct a dataset with two classes: each consisting of 16 tokens, where in one class, the final 4 tokens form a word. We identify specific heads in the early layers of models that contribute solely to the construction of these multi-token words. As discussed in (21), these heads are called "subjoiner" heads. As illustrated in Figure 9, layer 2 head 5 of Pythia 2.8B moves information from earlier tokens to the final token of the word. The attention heads



((a)) ((b))

Figure 10. (a) Accuracy of various linear probes on predicting “ing" for the final token position. Probes are trained on prediction and suppression neuron activations, where ensembles (blue line) outperform individual neuron probes (scatter plot) suggesting “prediction ensembling" that sometimes outperforms the model top-1 accuracy (red dotted) (b) Suppression (top) and prediction (bottom) when the next token of a word ends in -ing.

exhibit a consistent pattern, where attention decreases as tokens approach the final word. Specifically, the final token of the word attends most strongly to the first token, a feature absent in the baseline (shown on the left in Figure 9). This suggests at least one of many mechanisms by which models integrate local context, occurring at higher density in the first half of the models.

6.2. Study 2: Predicting the suffix -ing

Neurons performing prediction ensembling must work in tandem to predict the next token - akin to voting or operating in superposition. This suggests that multiple neurons working together may form a better prediction of the next token than a single neuron. To find evidence of this mechanism, we create a balanced dataset of two classes: tokens that do or do not end with the final token of "ing", all preceded by a context of 24 tokens. We train linear probes on the activation of 32 of the most active prediction and suppression neurons, both individually and in groups. We identify these neurons as outlined by (30), and provide examples of these neurons in GPT-2 XL (Figure 10(b)).

Probing Results We train two types of probes on activations at the penultimate token position of the dataset. First, we train 32 *individual* neuron probes and measure the classification accuracy (-ing/no -ing). We compare individual probes trained with the top-k neurons of the most accurate neurons, depicted by the line in Figure 10(a). We also note the mean model accuracy when predicting a token. Probes trained on suppression neurons, shown in yellow, resulted in the highest quality individual probes and performed similarly to the model itself, depicted by the dotted red line in Figure 10(a). Top-k probes trained with prediction neurons demonstrate even better accuracy than the average model prediction accuracy. Nonetheless, an individual neuron probe performs worse than any top-k probe, suggesting a critical role for ensembling in next-token prediction.

7. Concluding Remarks

Model Mystery A mystery occurs in the final layer of the GPT model, as suggested by Figure 5. Through our finds and suggested by (54), there is attention to MLP transition in all models, except an anomalous attention spike in the final layers of GPT, indicated by the $\frac{|Attn|}{|MLP|}$. We speculate that this is caused by tied embedding and unembedding weights (W_E and W_U) during the training of GPT-2 models. By tying these weights, GPT might be going against the “duality" discussed above. Tied weights force the “input space" and “output space" to look identical. This tying might re-involve attention units as done in the early layers of all models; however, we leave this as a future avenue to explore.

Limitations and Future Work Our study does not identify the specific causes of differences between GPT and Pythia models, such as whether redundancy stems from dropout during training, structural variations in attention and MLP mechanisms, a greater number of layers, or a combination thereof. Furthermore, the study relies on aggregation over many tokens, which may average out effects that occur to specific token classes. Despite this, our findings suggest phases of inference and future investigations using Sparse Autoencoders (SAEs) may form a connection to provide more evidence for or against this hypothesis.

Contributions VL conceived and led the study, performed all the analyses, and drafted the paper. WG and MT contributed to the experimental design and analytical methodology, and provided critical revisions of the paper, with WG additionally assisting in paper writing.

Acknowledgments Eric Michaud, Josh Engels, Dowon Baek, Isaac Liao, and the rest of the lab for helpful feedback. Thank you to MIT Supercloud for providing the compute possible for the project

References

[1] Hritik Bansal, Karthik Gopalakrishnan, Saket Dingliwal, Sravan Bodapati, Katrin Kirchhoff, and Dan Roth. Rethinking the role of scale for in-context learning: An interpretability-based case study at 66 billion scale. *arXiv preprint arXiv:2212.09095*, 2022.

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

[3] 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.

[4] Yoshua Bengio, Geoffrey Hinton, Andrew Yao, Dawn Song, Pieter Abbeel, Yuval Noah Harari, Ya-Qin Zhang, Lan Xue, Shai Shalev-Shwartz, Gillian Hadfield, et al. Managing ai risks in an era of rapid progress. *arXiv preprint arXiv:2310.17688*, 2023.

[5] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10231–10241, 2021.

[6] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023.

[7] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[8] Nick Cammarata, Gabriel Goh, Shan Carter, Chelsea Voss, Ludwig Schubert, and Chris Olah. Curve circuits. *Distill*, 2021. https://distill.pub/2020/circuits/curve-circuits.

[9] Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015*, 2019.

[10] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

[11] Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. Analyzing redundancy in pretrained transformer models. *arXiv preprint arXiv:2004.04010*, 2020.

[12] Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space. *arXiv preprint arXiv:2209.02535*, 2022.

[13] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. Softmax linear units. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/solu/index.html.

[16] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

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

[18] Chun Fan, Jiwei Li, Xiang Ao, Fei Wu, Yuxian Meng, and Xiaofei Sun. Layer-wise model pruning based on mutual information. *arXiv preprint arXiv:2108.12594*, 2021.

[19] Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? *arXiv preprint arXiv:2310.17191*, 2023.

[20] Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R Costa-jussà. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.

[21] Javier Ferrando and Elena Voita. Information flow routes: Automatically interpreting language models at scale. *arXiv preprint arXiv:2403.00824*, 2024.

[22] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[23] Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*, 2023.

[24] Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*, 2022.

[25] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

[26] Rhys Gould, Euan Ong, George Ogden, and Arthur Conmy. Successor heads: Recurring, interpretable attention heads in the wild. *arXiv preprint arXiv:2312.09230*, 2023.

[27] Klaus Greff, Rupesh K Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. *arXiv preprint arXiv:1612.07771*, 2016.

[28] 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.

[29] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

[30] Wes Gurnee, Theo Horsley, Zifan Carl Guo, Tara Rezaei Kheirkhah, Qinyi Sun, Will Hathaway, Neel Nanda, and Dimitris Bertsimas. Universal neurons in gpt2 language models. *arXiv preprint arXiv:2401.12181*, 2024.

[31] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

[32] Wes Gurnee and Max Tegmark. Language models represent space and time. *arXiv preprint arXiv:2310.02207*, 2023.

[33] Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[35] Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. *arXiv preprint arXiv:2310.15916*, 2023.

[36] Dan Hendrycks, Mantas Mazeika, and Thomas Woodside. An overview of catastrophic ai risks. *arXiv preprint arXiv:2306.12001*, 2023.

[37] Stanisław Jastrzębski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. *arXiv preprint arXiv:1710.04773*, 2017.

[38] Robert Krzyzanowski, Connor Kissane, Arthur Conmy, and Neel Nanda. We inspected every head in gpt-2 small using saes so you don't have to. Alignment Forum, 2024.

[39] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

[40] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR, 2023.

[41] Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2023.

[42] R Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L Griffiths. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *arXiv preprint arXiv:2309.13638*, 2023.

[43] Callum McDougall, Arthur Conmy, Cody Rushing, Thomas McGrath, and Neel Nanda. Copy suppression: Comprehensively understanding an attention head. *arXiv preprint arXiv:2310.04625*, 2023.

[44] Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations. *arXiv preprint arXiv:2307.15771*, 2023.

[45] 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.

[46] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

[47] Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Circuit component reuse across tasks in transformer language models. *arXiv preprint arXiv:2310.08744*, 2023.

[48] Neel Nanda. Transformerlens, 2022.

[49] Neel Nanda, Senthooran Rajamanoharan, Janos Kramar, and Rohin Shah. Fact finding: Attempting to reverse-engineer factual recall on the neuron level, Dec 2023.

[50] Nostalgebraist. Interpreting gpt: The logit lens. https://www.alignmentforum.org/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020.

[51] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

[52] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[53] Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. In *International Conference on Machine Learning*, pages 27011–27033. PMLR, 2023.

[54] Ofir Press, Noah A Smith, and Omer Levy. Improving transformer models by reordering their sublayers. *arXiv preprint arXiv:1911.03864*, 2019.

[55] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

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

[57] Cody Rushing and Neel Nanda. Explorations of self-repair in language models. *arXiv preprint arXiv:2402.15390*, 2024.

[58] Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429, 2023.

[59] Ludwig Schubert, Chelsea Voss, Nick Cammarata, Gabriel Goh, and Chris Olah. High-low frequency detectors. *Distill*, 2021. https://distill.pub/2020/circuits/frequency-edges.

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

[61] Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. Linear representations of sentiment in large language models. *arXiv preprint arXiv:2310.15154*, 2023.

[62] Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.

[63] Alexandre Variengien and Eric Winsor. Look before you leap: A universal emergent decomposition of retrieval tasks in language models, 2023.

[64] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29, 2016.

[65] Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. Neurons in large language models: Dead, n-gram, positional. *arXiv preprint arXiv:2309.04827*, 2023.

[66] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[67] Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Do llamas work in english? on the latent language of multilingual transformers. *arXiv preprint arXiv:2402.10588*, 2024.

[68] Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

[69] Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.

[70] Minjia Zhang and Yuxiong He. Accelerating training of transformer-based language models with progressive layer dropping. *Advances in Neural Information Processing Systems*, 33:14011–14023, 2020.

A. Speculations of Duality

Our findings suggest that the second half of the model is, in some sense, dual to the first half. This was briefly suggested in (15) in the context of compound words being broken down token-wise in the early layer but rebuilt in later layers. Self-repair discusses erasure and anti-erasure pairs in the first half and second half of model (44; 57), as coupled attention heads in copy suppression (43). Zooming out, the first half of models develop complex representation and the second half has the means to clear it 6. As seen by our experiments, interference with the first layer of models is also analogous to the interference of the final layer 3. While our study only suggests this we leave it for future work to investigate this further.

A. Cosine Similarity Analysis of Swapped Layers

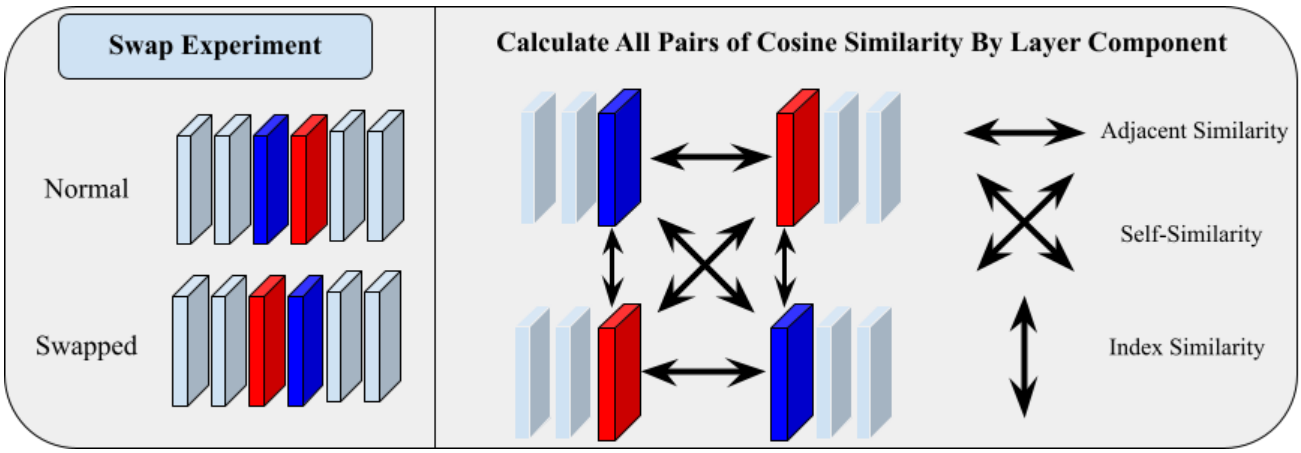


Figure 11. We compute pairwise cosine similarities between a standard operational model and a model with two adjacent layers swapped, analyzing the component-wise outputs (MLP and ATTN). This approach aims to explore three specific properties: *Adjacent Similarity*, which quantifies the similarity of component outputs to assess iterative inference; *Self-Similarity*, which evaluates the resistance of a layer to change when relocated, serving as a measure of layer “stubbornness"; and *Index Similarity*, which examines the adaptability of a layer in a new position, indicating layer “flexibility."

Cosine Similarity Metrics We collect three key metrics to compare a normal LLM to one with a set of adjacent layers swapped. First, *self-similarity* measures how much a layer retains its function after a swap, reflecting its "stubbornness." A high self-similarity score indicates that a layer continues to project similar contents to the residual stream, even after its position in the network has been changed. Second, *index similarity* assesses how closely the output of a swapped layer matches the output of the original layer it replaced. This metric serves as an indicator of a layer’s flexibility, with a high score suggesting that the layer can effectively assume the computational role of its predecessor, which could range from active processing to merely acting as a pass-through in the network. Lastly, adjacent similarity provides a baseline comparison by measuring the similarity in computations between adjacent layers in an unmanipulated model. This metric helps establish how similar or diverse the functions of neighboring layers are under normal conditions. By comparing these metrics across different stages of inference, we can gain insights into the commutativity of layers and the nature of the computations performed at each stage.

Cosine Similarity Results Here we focus on Pythia 1.4B and GPT-2 XL, which contain a similar number of parameters (1.4B and 1.5B respectively). GPT-2 displays smoother trends compared to its Pythia counterpart while exhibiting similar overall patterns. We hypothesize that this is a result of differences in training dynamics (e.g., the use of dropout in GPT-2) and the fact that the GPT-2 model contains more layers. A larger number of layers presents greater opportunities for manipulating the output distribution and allows for more gradual changes. From an optimization perspective, this is analogous to taking smaller but more frequent gradient steps. Increasing the number of layers may also provide a means for greater redundancy in models, a key feature of GPT-style models that we discuss further below.

As seen in Figure 12, all model components maintain high degrees of self-similarity (denoted by the blue and red lines), suggesting that a component’s position does not significantly affect how it projects onto the residual stream when swapped. This finding has implications for how we interpret the remaining metrics. Another commonality across all plots is a

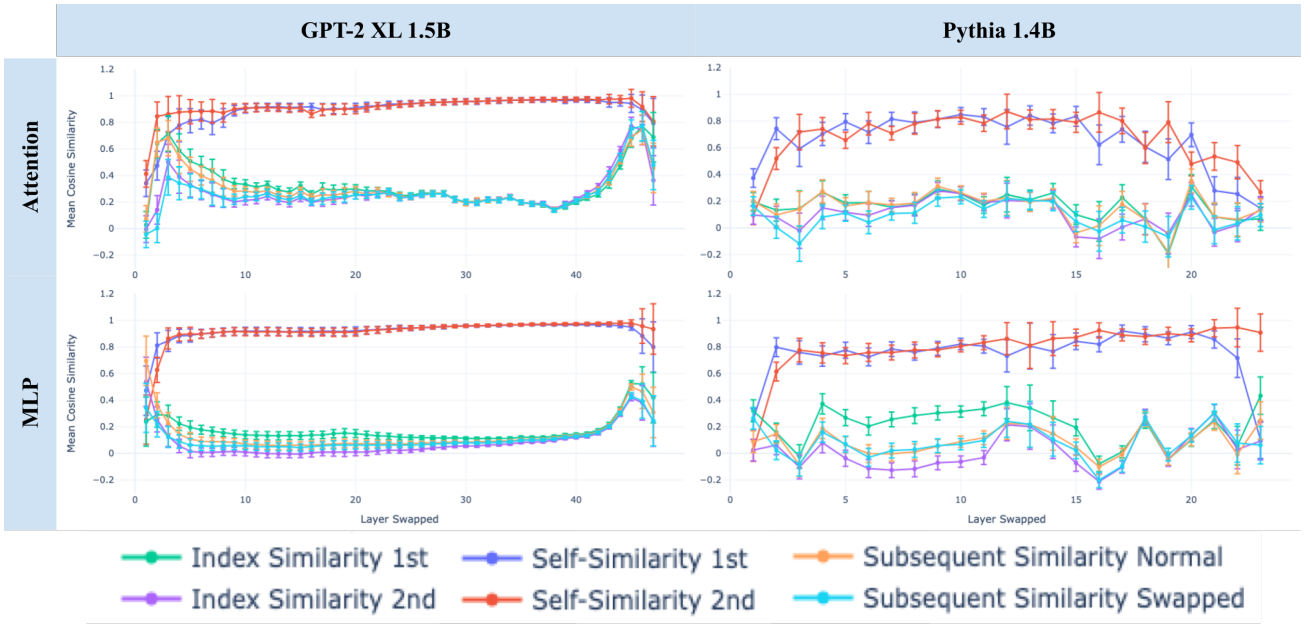


Figure 12. We compute pairwise cosine similarities between a standard operational model and a model with two adjacent layers swapped, as depicted in 11, across two different models. w high index similarity, marked by the teal line, suggests that when a layer is moved earlier in the computational sequence, it retains a similar projection onto the residual stream as the layer it replaced. This observation supports the concept of iterative inference, highlighting overlapping computational roles between adjacent layers.

significant change in metrics approximately halfway through the model, which we interpret as the separation between stages 2 and 3. Specifically, we observe a sharp decrease in index similarity and an increase in orthogonality between the swapped layer and its neighbors, suggesting a transition from iterative refinement to more specialized computations.

Attention Heads Both models exhibit distinct patterns in attention-head behavior in the latter half of the network. In Pythia models, the attention head metrics converge to orthogonality, while in GPT-2 models, they converge to similarity. For Pythia, the self-similarity of attention heads decreases, indicating that they become less "stubborn" and more sensitive to their position in the network. In contrast, attention heads in GPT-2 models become increasingly redundant, with high self-similarity and index similarity scores. We hypothesize that this increased redundancy arises from the larger number of layers in GPT-2 models, which allows for a more gradual refinement of the output distribution. This finding has important implications for model design, suggesting that there may be an optimal number of layers given total parameters to balance computational efficiency and redundancy.

MLPs The MLP components display two significant patterns across models. First, in the region corresponding to stage 2 of inference, we observe that the index similarity (teal line) is higher than both the adjacent similarity and the self-similarity scores. This pattern provides evidence for iterative inference, where a layer moved earlier in the computation has a projection onto the residual stream that overlaps more strongly with its previous neighbor than with its original position or its new neighbor. This overlap is more pronounced in Pythia models than in GPT-2 models, possibly because Pythia models have fewer layers to complete stage 2 of inference.

Second, in stage 3 of inference, both models demonstrate a convergence of all metrics except self-similarity toward orthogonality. The combination of high self-similarity (indicating stubbornness) and orthogonality to the replaced layer and the adjacent layers suggests a high degree of specialization in the MLPs of stage 3.

D. Additional Empirical Details

All experimental code for future experiments is available at:
https://github.com/vdlad/Remarkable-Robustness-of-LLMs.

B. Single Component Ablations Pythia 1.4B and GPT-2

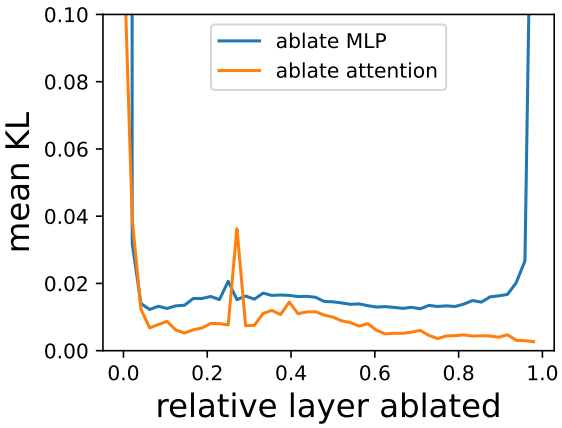


Figure 13. Ablate GPT-2 XL

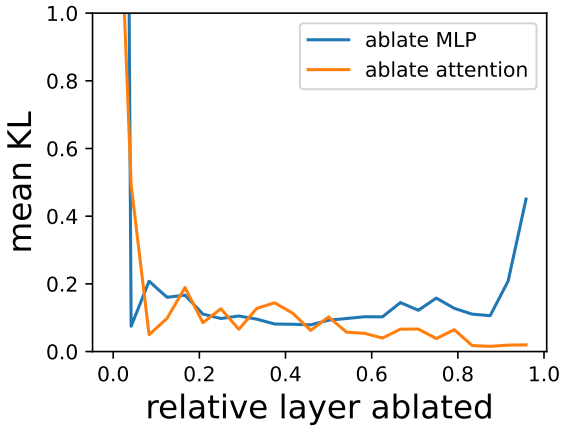


Figure 14. Ablate Pythia 1.4B

Figure 15. Ablating the MLP in both models increases the KL divergence between the nominal and ablated models in the final layers, suggesting neuron dependencies in later layers.

Name	HuggingFace Model Name
Pythia 410M	EleutherAI/pythia-410m-deduped
Pythia 1.4B	EleutherAI/pythia-1.4b-deduped
Pythia 2.8B	EleutherAI/pythia-2.8b-deduped
Pythia 6.9B	EleutherAI/pythia-6.9b-deduped
GPT-2 Small (124M)	gpt2
GPT-2 Medium (355M)	gpt2-medium
GPT-2 Large (774M)	gpt2-large
GPT-2 XL (1.5B)	gpt2-xl
Phi 1 (1.3B)	microsoft/Phi-1
Phi 1.5 (1.3B)	microsoft/Phi-1.5
Phi 2 (2.7B)	microsoft/Phi-2
The Pile	EleutherAI/the_pile_deduplicated

Table 3. List of models and dataset used in the experiments.

We make ubiquitous use of TransformerLens (48) to perform hooks and transformer manipulations.

For specificity, we utilize the following HuggingFace model names, and dataset. We do not change the parameters of the models from what they are described on the HuggingFace page.

All experiments described can be performed on a single NVIDIA A6000. We utilized 2 NVIDIA A6000 and 500 GB of RAM. To aggregate the metrics described in the paper, we run the model on 1 million tokens ℓ times, where ℓ is the number of layers. This takes on average 8 hours per model, per layer intervention (swapping and ablating). We save this aggregation for data analysis.

We utilize several conventional weight preprocessing techniques to streamline our calculations (48).

Layer Norm Preprocessing Following (30), before each MLP calculation, a layer norm operation is applied to the residual stream. This normalizes the input before the MLP. The TransformerLens package simplifies this process by incorporating the layer norm into the weights and biases of the MLP, resulting in matrices W_{eff} and b_{eff}. In many layer norm implementations, trainable parameters $\boldsymbol{\gamma} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^n$ are included:

$$\text{LayerNorm}(\mathbf{x}) = \frac{\mathbf{x} - \mathbb{E}(\mathbf{x})}{\sqrt{\text{Var}(\mathbf{x})}} * \boldsymbol{\gamma} + \mathbf{b}. \tag{1}$$

C. Top Prediction and Suppression Neurons

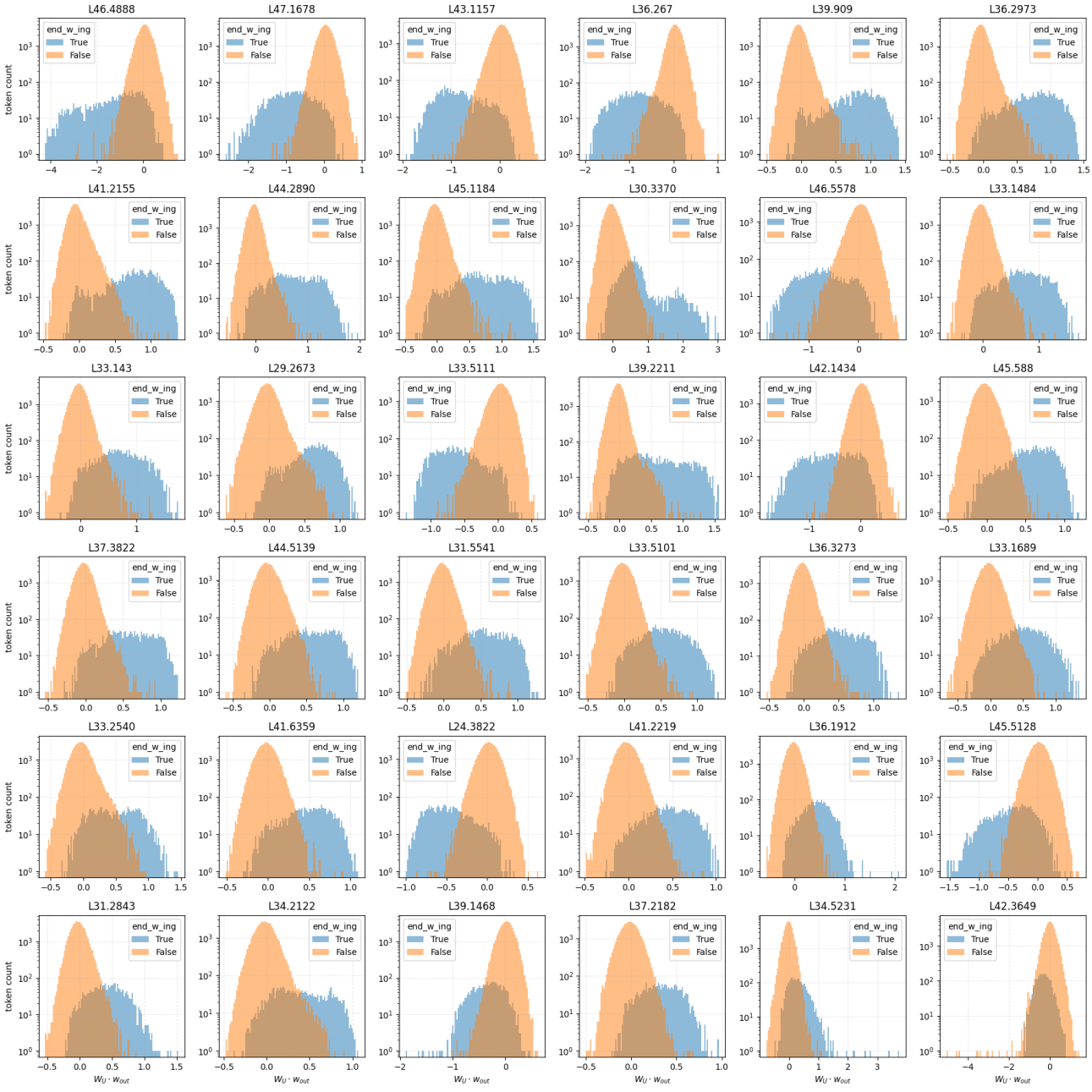


Figure 16. Top 36 prediction and suppression neurons for -ing which have the greatest mean absolute difference between respective ($W_U \cdot w_{\text{out}}$). This is the product between the model unembedding weights and output weights of MLP.

We "fold" the layer norm parameters into W_{in} by treating the layer norm as a linear layer and then merging the subsequent layers:

$$\mathbf{W}_{\text{eff}} = \mathbf{W}_{\text{in}}\, \mathbf{diag}(\boldsymbol{\gamma}) \qquad \mathbf{b}_{\text{eff}} = \mathbf{b}_{\text{in}} + \mathbf{W}_{\text{in}}\mathbf{b} \tag{2}$$

Additionally, we then center reading weights. Thus, we adjust the weights $\mathbf{W}_{\text{eff}}$ as follows:

$$\mathbf{W}^{'}_{\text{eff}}(i,:) = \mathbf{W}_{\text{eff}}(i,:) - \bar{\mathbf{W}}_{\text{eff}}(i,:)$$

Centering Writing Weights Because of the LayerNorm operation in every layer, we can align weighs with the all-one direction in the residual stream as they do not influence the model's calculations. Therefore, we mean-center $\mathbf{W}_{\text{out}}$ and $\mathbf{b}_{\text{out}}$ by subtracting the column means of $\mathbf{W}_{\text{out}}$:

$$\mathbf{W}^{'}_{\text{out}}(:,i) = \mathbf{W}_{\text{out}}(:,i) - \bar{\mathbf{W}}_{\text{out}}(:,i)$$