EVERY COMPONENT IS A LOOKUP: ONE LINEAR GRAPH FOR INTERACTION, COMPOSITION AND ATTRIBUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Interpretability methods for transformers are typically built around separate questions: which components interact, how information routes to the output, and which input tokens contribute. Because these methods rely on different assumptions, their answers are difficult to relate. We argue that two architecturally motivated assumptions suffice to address all three questions: **attention and MLPs share a key-value form**, $\phi(S)\,U$, in which $\phi(S)$ selects over values U, and **components read from an additive residual stream**, the sum of component outputs. Holding these selections at their forward-pass values turns the model into a computational graph, of which component interactions, composition paths, and token attribution are different readouts. We develop UNPACK, a backward attribution procedure over this graph, and validate each readout against the corresponding established test: interaction scores predict ablation effects across models from 160M to 6.9B parameters, recovered routes reproduce established circuits down to the key, query, or value branch the circuit specifies, and token attribution passes the same faithfulness test as dedicated attribution methods. The results suggest that these two assumptions suffice for the interpretability questions above. On a task with a known circuit, we find that contribution and causal effect can differ, and that the difference has a recognisable signature: components that matter for the task change their contribution when the task is removed from the input, while components that act like a bias term do not.

1 INTRODUCTION

In mechanistic interpretability, we are interested in three main questions about a single prediction. (1) Which components feed which, and how strongly? (2) How do those connections compose into the route that produced the answer? (3) Which input tokens drove that answer, and which pushed against it? Each question currently has its own tool, and each tool its own assumptions: path patching establishes which components feed which by intervening with a designed counterfactual prompt (Wang et al., 2023), Information Flow Routes trace the edges between components through a chosen proximity rule (Ferrando & Voita, 2024), and LRP methods score the input tokens through heuristic propagation rules (Bach et al., 2015; Achtibat et al., 2024). In this paper we show how a single linear graph can unify these three views.

The separate assumptions are problematic. Take the LRP family for example: different propagation rules give different attribution scores, so one can neither say what a score means nor which rule is better. And the problem compounds across methods: a method that scores tokens and one that scores components cannot be compared at all; There is no common quantity that both methods measure. Interventional methods avoid this problem because a patched effect has a causal interpretation, but they require a specific counterfactual for each question, and not every prompt supports one.

Motivated by these cases, we argue that two architecturally motivated assumptions suffice to answer all three questions from a single computational graph: both MHA and MLP follow the same key-value template, and both read from a linear residual stream. Self-attention computes $\phi(S)\,U$, a selection $\phi(S)$ over values U, with bilinear scores $S = QK^\top/\sqrt{d_{\text{head}}}$, while an MLP has the same form, with the up-projection's columns as keys, the activation as selection, and the down-projection's

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

rows as values (Geva et al., 2021). Because the residual stream is a sum of component outputs (Elhage et al., 2021), holding the routing weights $\phi(S)$ fixed at their forward-pass values makes every score decompose additively over the components that wrote the stream. This yields a direct linear connection between any pair of components, turning the model into a computational graph: connection strengths answer *which components feed which*, paths answer *how those connections compose*, and the credit the paths deposit at the input embeddings answers *which inputs support or oppose the prediction*. The two assumptions determine the structure of this graph completely; the remaining choices of credit normalization and dispatch are stated in Section 5, and Appendix I shows the results are stable across them.

We develop this view into an algorithm, UNPACK (**Uni**fied **P**ath **A**ttribution through **C**omponent **K**eys), a backward walk over the graph on a single forward pass, with nothing patched, no gradients, and no training, and validate its three outputs against the external checks each purpose-built tool is trusted for: the interaction scores against causal ablation across the Pythia-deduped family (Section 4.1), the routes against two circuits established by path patching (Section 4.2 and Appendix F), and the token attribution against an input-flipping perturbation check (Section 4.3). The graph reads out contribution under the realized computation; Section 5 discusses where contribution and causal effect diverge. Our contributions are:

- **A claim and its validation.** We argue that two architecturally motivated assumptions, the shared key-value template and the linear residual stream, are sufficient to answer all three questions from a single computational graph. We validate each readout against its purpose-built counterpart: interaction scores predict causal ablation effects across the Pythia-deduped family (within-layer Spearman $\rho = 0.72$–0.96), routes recover two circuits established by path patching down to the key, query, or value branch the circuit names, and token attribution passes the same perturbation check as dedicated attribution methods, using identical hyperparameters throughout.
- **An algorithm.** UNPACK constructs the graph from a single forward pass: no patching, gradients, or training are required. The graph can be rerooted at any component, providing all three readouts from one run.
- **Findings.** Observational credit alone does not give causal effect: layer-0 components receive high credit on every prompt, task or not, and this only becomes visible by comparing against a matched counterfactual, on which credit to the causally established components collapses while layer-0 credit stays put. The credit nonetheless carries mechanism, not just magnitude: on IOI, token attribution tracks a token's circuit role rather than its identity, as the recovered paths predict, and this holds from 160M to 6.9B parameters.

2 BACKGROUND AND RELATED WORK

Setting and notation. A decoder-only Transformer (Vaswani et al., 2017) stacks L blocks, each with a multi-head self-attention (MHA) sublayer and an MLP sublayer, both reading from and writing to a residual stream of width d_{model}. Head h at query position q forms weights $\alpha_{h,q,s}$ by softmax over the scaled scores $\langle X_q, X_s\rangle_{\text{QK}} := (X_q W_Q)(X_s W_K)^\top / \sqrt{d_{\text{head}}}$ and writes $\sum_s \alpha_{h,q,s}\, X_s W_V W_O$; the MLP applies $W_{\text{down}}\, \phi(W_{\text{up}} X_q)$ at each position. We assume the pre-norm placement used by GPT-2 and Pythia, with LayerNorm on each sublayer's input.

Residual stream. Each sublayer adds its output to the stream, so the residual state at any layer is a sum, $X_j = \sum_k c_{kj}$, where each c_{kj} is the output of one component at position j: an attention head, an MLP, or the input embedding (Elhage et al., 2021). Each component's contribution is a vector in $\mathbb{R}^{d_{\text{model}}}$ that can be examined on its own, which is the property post-hoc interpretability of transformers generally rests on.

Token attribution. Token attribution asks, for a single prediction, how much each input token contributed to it: one score per input position, signed if the method distinguishes tokens that pushed the prediction up from tokens that pushed it down. Gradient- and attention-based methods include integrated gradients and attention rollout (Sundararajan et al., 2017; Abnar & Zuidema, 2020); relevance-based methods redistribute a conserved score backward through the network by

per-layer rules (Bach et al., 2015; Chefer et al., 2021; Ali et al., 2022). Section 4.3 runs our token readout through the input-flipping check these methods are evaluated on, alongside three of them: AttnLRP (Achtibat et al., 2024), which derives propagation rules through the softmax and the bilinear attention product; its variant CP-LRP, which detaches the gradient through the attention pattern; and Information Flow Routes (IFR; Ferrando & Voita, 2024), which propagates L_1 mass backward through the residual stream. AttnLRP and CP-LRP are signed; IFR's per-token scores are non-negative by construction and do not depend on the target token.

Component attribution and interactions. The same question is asked of the model's internal parts: how much did each head or MLP contribute, and which component feeds which through which of its inputs. Direct logit attribution (nostalgebraist, 2020) projects each component's output onto the target unembedding direction; the importance our recursion starts from (Eq. 4) is this quantity, which the recursion then passes backward. Interactions are at present established by intervening: path patching (Wang et al., 2023) overwrites the activation on one edge with its value on a counterfactual prompt and measures the effect downstream, and this is how the IOI connections of Section 4.2, and the branch each one uses, were established. Automated and trained variants exist (Conmy et al., 2023; Hanna et al., 2024; Ameisen et al., 2025).

3 METHOD

3.1 TWO ASSUMPTIONS: LINEARITY AND UNIFIED KEY-VALUE FRAMEWORK

Our method rests on two structural properties of Transformers: *linearity* in how sublayer inputs are composed, and a shared *key-value lookup* structure across MHA and MLPs.

First, the residual stream (Elhage et al., 2021) is additive over the outputs of upstream components, $X_j = \sum_k c_{kj}$. The downstream components read it with a linear or bilinear function. For MLP we have

$$X_q W_{\text{up}} = \sum_k c_{kq} W_{\text{up}}. \tag{1}$$

For MHA we have

$$\langle X_q, X_s \rangle_{\text{QK}} = \sum_k \langle X_q, c_{ks} \rangle_{\text{QK}} = \sum_k \langle c_{kq}, X_s \rangle_{\text{QK}}. \tag{2}$$

Second, both MHA and MLPs can be written in a common key-value form (Figure 1) following the interpretation of Geva et al. (2021). Inside a component, the residual representation X matches each of the keys and produces the score S_i, this score is then fed into the nonlinear function ϕ to become the weight to select the value U_i. Consequently, we can represent both MHA and MLP in one format (Table 1):

$$\text{Sublayer}(X) = \phi(S)\, U. \tag{3}$$

With these assumptions, we can view a transformer model as a computational graph. Between two components, one interacts with the other by contributing to the key-scores S_i. Inside a component, each S_i decides how much the corresponding value U_i is selected. To trace the computation backwards, we can reverse the aforesaid order: we first decide the credit of each value U_i, the credit is handed to the corresponding score S_i, the same credit is then further dispatched among upstream components feeding S_i. Section 3.2 turns this idea into an algorithm and Section 3.3 shows how this framework and algorithm can answer multiple interpretability questions together.

3.2 RECURSIVE BACKWARD ATTRIBUTION

This algorithm recursively dispatches the credit of the target logit backward, following the mental model of Section 3.1. Stage 1 and Stage 2 correspond to the key-value and then linearity assumptions. We describe the root and the two stages here; the full algorithm is in Appendix A.

Setup. At the target position p, each residual component c_k has a scalar contribution to the target logit, obtained by projecting c_k onto the target unembedding direction:

$$I(c_k) = \langle c_k(p),\, d_t \rangle, \qquad d_t := \frac{w_{\text{LN}}}{\sqrt{\text{Var}(X_p) + \epsilon}} \odot \big(W_U[t] - \overline{W}_U\big), \tag{4}$$

Table 1: Unified key-value format $\phi(S)\,U$ for attention and MLP. The entries i are what the sum $\sum_i \phi(S)_i U_i$ runs over, and are what the recursion of Section 3.2 dispatches credit to.

	Attention	MLP
ϕ	softmax	ReLU / GELU
S	$QK^T/\sqrt{d_{\text{head}}}$	XW_{up}
U	XW_V	W_{down}
entries i	source positions s	neurons j

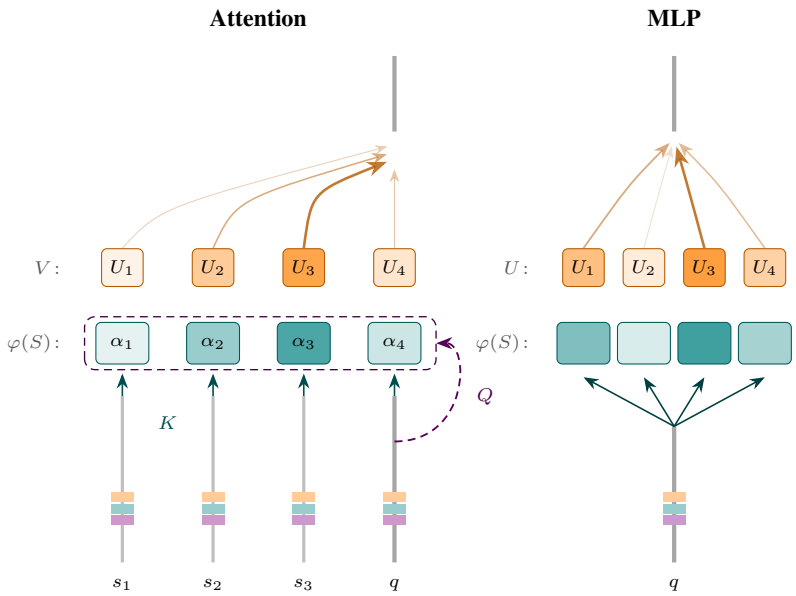


Figure 1: Unified key-value view. Both attention and MLP compute a weighted sum $\sum_i \varphi(S_i) \cdot U_i$: a selection mechanism $\varphi(S)$ (green, opacity indicates strength) weights values U (orange), and the result is added to the residual stream at position q. In attention, position q serves as both query and key; K and V are indexed by all positions including q, so the output aggregates across positions, while Q reads from q alone. In MLP, S and U are indexed by neuron, so all computation stays at q. Colored blocks at each position represent upstream residual components c_k; since S decomposes additively across these components, backward credit attribution traces the same paths in reverse.

where $\overline{W}_U$ is the vocabulary-averaged unembedding vector.[1] We call $I(c_k)$ the component's *credit*; it is what the recursion redistributes.

Stage 1: credit to the entries. The entries i indexing $\phi(S)\,U$ are source positions for a head and neurons for an MLP. Credit I arriving at a component is divided among them by the part of the output each one supplied along the direction the credit is travelling in:

$$I_i \;=\; I \cdot \frac{\phi(S)_i \,\langle U_i,\, d\rangle}{\textsc{SafeDenom}_k\big(\phi(S)_k \,\langle U_k, d\rangle\big)}, \qquad d \;=\; \begin{cases} d_t & \text{at the target (Eq. 4),} \\ \phi(S)\,U & \text{deeper in the recursion.} \end{cases} \tag{5}$$

An entry whose value pushes along d takes positive credit, one that pushes against it takes negative credit, and an unselected entry takes none. SAFEDENOM (App. A.1) floors the denominator so that credit is not amplified without bound when positive and negative terms cancel.

Stage 2: credit from an entry to the components that built it. Two things built entry i: the score S_i that selected it, and the value U_i it holds. Each is additive over the upstream components that

[1]The centering makes $I(c_k)$ measure how much c_k raises the target token's probability rather than its raw logit, which softmax identifies only up to a constant.

wrote the stream (Section 3.1), so I_i is divided among those components in proportion to their contributions, again through SAFEDENOM. The two stages then repeat at each component so credited, until the credit reaches the embeddings.

For an MLP, the entries are neurons and $U_j = W^j_{\text{down}}$ is a parameter, so only the score carries upstream credit: Stage 2 has a single branch, the writers of the pre-activation at the MLP's own position.

An attention head reads both S and U from the stream, at two different positions, so Stage 2 splits three ways, each branch following one split of the head's inputs:

- the *key* branch divides I_s by the key-side split of Section 3.1 and continues at s;
- the *query* branch divides I_s by the query-side split and continues at q;
- the *value* branch divides I_s by the writers' shares of $U_s = X_s W_V W_O$, itself linear in the stream, and continues at s.

These are the Q-, K- and V-composition modes of Elhage et al. (2021). We follow all three with equal weight ($w_K = w_Q = w_V = \frac{1}{3}$) and label each hop with the branch it took; Appendix A.2 gives the three scores in full, Figure 1 draws the three routes, and Section 4.2 reads the labels out.

Termination and conservation. Source layers strictly decrease at every recursion step, so every path terminates at the embedding layer, where credit accumulates at the token position. Aggregate credit is conserved up to the soft floor on the denominators (Appendix A.1), which prevents amplification when positive and negative contributions nearly cancel. We use $\beta = 0.8$ throughout. When enumerating named paths, we drop sub-credits whose magnitude falls below a pruning threshold $\tau = 10^{-3}$; token attribution is computed exactly and does not depend on τ (Appendix A.3).

3.3 THREE READOUTS OF THE GRAPH

The graph is built once per forward pass. Everything the paper reports is read from it in one of three ways: edge weights between components, paths through the graph, and per-token credit at the input.

Component interaction scores. Based on the framework, we can define the interaction score between two components as how strongly one affects the other's selection scores. For MHA the score is a row over source positions s. Supposing the downstream head is component m with query position q, we define the score as:

$$\text{score}^{\text{attn}}_{k\to m} = \text{std}(\{\langle X_q, c_{ks}\rangle_{\text{QK}}\}_s), \tag{6}$$

since softmax is translation-invariant and only the spread of a row moves the attention pattern. For an MLP m at position q, the activations are magnitude-sensitive and both signs of a pre-activation are meaningful, so we use the L2 norm,

$$\text{score}^{\text{mlp}}_{k\to m} = \|c_{kq} W_{\text{up}}\|_2. \tag{7}$$

These are the edge weights of the graph; Section 4.1 checks them against ablation.

Named composition paths. Recording the path each unit of credit takes from the unembedding back to an embedding gives one path of how the token contributes to the target logit. Each hop names the component and, at attention, the K, Q, or V branch; the credit reaching the embedding scores the path, e.g. `emb@S2` $\to$ `MLP0@S2` $\xrightarrow{V}$ `A8.H6@END` (Figure 5). Ranking paths by credit gives the dominant composition routes for a prediction. Since the decomposition is defined at every component, the recursion can be rooted at any of them instead of the target logit (*rerooting*), enumerating the paths into that component.

Token attribution. Summing the credit that reaches each input token, and normalizing by the total positive credit, gives a signed score per token: positive if the token pushed the target up, negative if it pushed it down. This is the same quantity as the paths above, summed over all paths ending at each token.

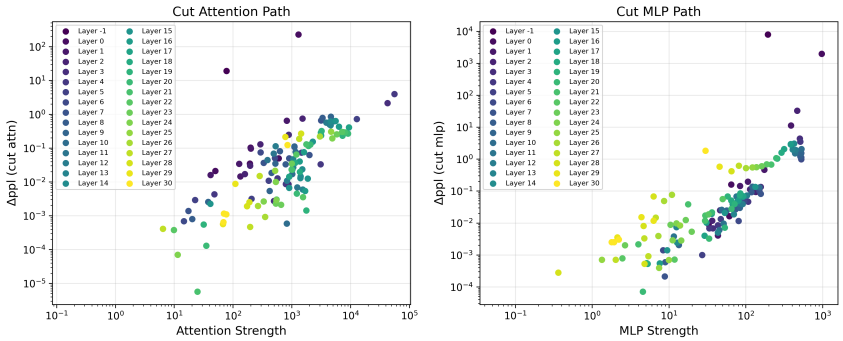


Figure 2: Interaction scores against causal knockout effects (Pythia-6.9B-deduped, 156 knockouts). Each point is one residual-stream component; the x-axis is its aggregated score, the y-axis is Δppl when the corresponding communication channel is ablated. **Left:** attention channel. **Right:** MLP channel. Points are colored by source layer (darker = earlier; layer -1 is the embedding); both axes log-scaled. Within-layer Spearman $\rho = 0.86$ / 0.90; the same pattern holds from 160M to 6.9B (Appendix C).

4 Experiments and Results

Do the two assumptions suffice for interpretability tasks? We validate this claim by checking each readout of Section 3.3 against an established method: intervention for the edge weights (Section 4.1), circuits established by path patching for the paths (Section 4.2), and input perturbation for the token credit (Section 4.3). The decomposition, its hyperparameters, and the branch weights are identical across all three checks.

4.1 Component interaction scores predict knockout effects

We test whether the interaction scores of Section 3.3 predict what happens when a component's communication is removed. Over 1,000 Pile sentences (Gao et al., 2020), every residual-stream component receives two aggregated strengths: its summed key-side pull on all downstream attention heads, and the analogous quantity on all downstream MLPs. The ablation is communication-specific: the component's residual vector c_k, extracted on a clean forward pass, is subtracted from the input of one type of downstream LayerNorm (cut-attention or cut-MLP), and we measure the change in perplexity (Δppl) on held-out Pile text. Formulas, sampling, and per-model counts are in Appendix C.

Figure 2 shows the result for Pythia-6.9B-deduped (156 knockouts): the score-to-ablation relationship is strongly monotonic on both channels, spanning four to five orders of magnitude on each axis, with within-layer Spearman $\rho = 0.86$ (attention) and 0.90 (MLP). The same pattern holds across the Pythia-deduped family (Biderman et al., 2023) from 160M to 6.9B parameters, with within-layer ρ between 0.72 and 0.96 at every scale (Appendix C).

4.2 Composition

Wang et al. (2023) identified a 26-head circuit in GPT-2 small for the IOI task, which asks the model to complete sentences such as "Alice and Bob went to the store. Bob gave a drink to" with the name that occurs once ("Alice"). Within the circuit, they name three connections together with the branch each travels: S-Inhibition heads, which suppress the repeated name, modulate the *queries* of Name Mover heads, which copy a name to the output (their Sec. 3.2); Duplicate Token and Induction heads, which detect the repetition, write positional signal into S-Inhibition *values* (Sec. 3.3); and Previous Token heads feed Induction *keys* (Sec. 3.3). We test the compositions found with our method against these claims: over 100 prompts, one decomposition per prompt, we reroot at the head of each claim[2] and rank upstream attention heads of layer ≥ 1 by credit in the claimed branch.

[2]At END for the first two claims, at S2 for the third.

Table 2: Composition verification via rerooting on IOI (GPT-2 small, $n = 100$). Cells: median rank / found%, ranking upstream attention heads (layer ≥ 1) by credit in the branch of that row. Bold marks the connection claimed by Wang et al. (2023). The first two rows reroot at the same Name Mover heads and differ only in the branch, isolating the mode. Pooled over the heads of each role ($n = 300$ for 3 Name Movers, $n = 400$ for 4 S-Inhibition or 4 Induction heads).

Branch	Root	NM	S-Inh	Ind	Dup	Prev
K	NM	–	10 / 48%	14 / 27%	13 / 19%	14 / 46%
Q	NM	2 / 33%	**1 / 100%**	4 / 63%	18 / 5%	10 / 56%
V	S-Inh	–	–	**2 / 96%**	**1 / 95%**	10 / 21%
K	Ind	–	–	21 / 3%	8 / 10%	**3 / 98%**

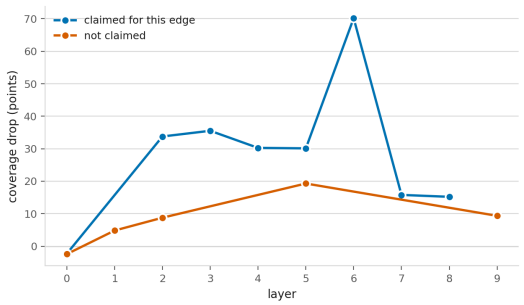


Figure 3: Coverage drop from IOI prompts to matched ABC controls (GPT-2 small), per source layer, pooled over the three claimed edges of Table 2: how much less often a component appears among the upstream feeders of the claimed branch, re-rootings paired across arms. Controls are the prompts on which the model does not rank A as the top prediction; the claimed line keeps all claimed components, the unclaimed line, the ten strongest unclaimed ones. Claimed components lose 30–70 points at layers 2–6 and about 15 at layers 7–8, unclaimed at most 19; at layer 0 both groups barely move, behaving like a bias term. Details in Appendix E.

We report the median rank of the expected upstream role and found%, the fraction of rerootings in which it appears at all.

These connections were established by path patching, which measures the causal effect of ablating an edge on a counterfactual prompt; our method is observational, and a component can contribute much while being causally unnecessary, so the two need not agree. However, all three connections surface at the top of the ranking (Table 2): in the branch each is claimed to use, the named upstream role appears in at least 95% of rerootings with median rank at most 3. The routing is also branch-specific: at the same Name Mover heads, S-Inhibition is the strongest Q-side input (rank 1, 100%) but rank 10 on the K side, the distinction Wang et al. (2023) draw and one no single per-token or per-component score can express. Full matrices are in Appendix D. Appendix F repeats the check on the greater-than circuit of Hanna et al. (2023).

Some ranks in Table 2 worsen once layer-0 components are included (Appendix D). We hypothesize that layer-0 components act like a bias term: they contribute to every prompt, whether or not the task is running. To check this, we track whether each component still appears among the upstream feeders of the claimed branch when moving from the IOI prompts to the matched ABC controls, and record the *coverage drop*: how much less often it appears when the task is absent (Figure 3). The components found by path patching drop far more in coverage than the others: their credit is tied to the task. Layer-0 components instead stay put in both prompt sets, keeping their high ranks on any prompt, as a bias would.

4.3 Token attribution

We evaluate the faithfulness of token attribution with the input flipping protocol of Achtibat et al. (2024), reporting Information Flow Routes (IFR), AttnLRP and CP-LRP for comparison. The pro-

Table 3: Perturbation faithfulness on 300 Pile sentences (GPT-2 small). ΔA is the area between the LeRF and MoRF curves in probability units, and %f the fraction of sentences with $\Delta A > 0$ (50% is chance).

Method	ΔA	%f
UNPACK	$+0.085$	100
AttnLRP	$+0.066$	99
CP-LRP	$+0.049$	91
IFR	$+0.084$	99
Random ranking	-0.001	49

tocol zeroes input embeddings cumulatively, in most-relevant-first (MoRF) and least-relevant-first (LeRF) order, and scores the area ΔA between the two target-probability curves (LeRF minus MoRF); %f is the fraction of sentences with $\Delta A > 0$ (50% is chance). All four methods clear the check (Table 3; setup and target direction in Appendix G); UNPACK reaches the largest ΔA (0.085) on 100% of sentences, against -0.001 (49%) for a random ranking.

Intuitively, our token attribution measures the sum of contributions of all paths connecting one input token to the target logit. Since each path records which components compose and through which branch, the token attribution inherits this mechanism. To demonstrate this, we consider the IOI task again. According to Wang et al. (2023), the S2 token (the position where one of the names occurs a second time) triggers the duplicate detection, and the S-Inhibition heads read from it to suppress that name at the final position.

A naive check for suppression is whether S2's credit for predicting S is negative. However, no other token in the template (the indirect object, the verb, the object) has an obvious reason to contribute to predicting S, so credit for S lands mostly on S1 and S2, and S2's credit may stay positive simply because of this copying. We therefore design a comparative experiment to separate the suppression mechanism from the copying mechanism. We do token attribution for both IOI and the matched ABC controls of Section 4.2, in which no name repeats. Let $\mathrm{X} \to \mathrm{Y}$ denote the credit to token X when the target is Y with a contrastive distractor per condition. For IOI we report $\mathrm{S1} \to \mathrm{S}$ and $\mathrm{S2} \to \mathrm{S}$, the credits to two occurrences of the same token in one prompt. For ABC we report $\mathrm{C} \to \mathrm{C}$, the credit a name gets for predicting itself with nothing to suppress it (pure copying), and $\mathrm{C} \to \mathrm{B}$, the credit the same slot gets for an unrelated name (position alone). We also report a normalized score $E = 1 - \frac{\mathrm{S2}\to\mathrm{S} - \mathrm{C}\to\mathrm{B}}{\mathrm{C}\to\mathrm{C} - \mathrm{C}\to\mathrm{B}}$, the fraction of copying credit removed at S2: $E \approx 1$ means suppression erased it, $E \approx 0$ that it survived, allowing comparison across methods.

Table 4 shows the results. All methods capture some suppression signal: every $\mathrm{S2} \to \mathrm{S}$ value is smaller than the corresponding $\mathrm{C} \to \mathrm{C}$ and $\mathrm{S1} \to \mathrm{S}$. IFR shows no copying signal on the ABC prompts ($\mathrm{C} \to \mathrm{C} = \mathrm{C} \to \mathrm{B}$), so E is undefined for it. Our method shows the strongest suppression: its $\mathrm{S2} \to \mathrm{S}$ falls below even its own positional baseline $\mathrm{C} \to \mathrm{B}$, and its E exceeds that of AttnLRP and CP-LRP ($E > 1$: S2 receives less credit than the position earns for an arbitrary name).

Note that there is no ground truth for this comparison: the methods differ in their propagation rules, so a higher E does not mean a better method, and we do not claim to beat any baseline in Table 4. This is precisely what the unified view adds. Because our credit follows from the two assumptions alone, the numbers have a definite reading: the $\mathrm{S2} \to \mathrm{S}$ column measures paths from S2 that, through the key–value interactions, end up suppressing the prediction of its own name. The same paths can be inspected directly (Section 4.2), and they name the mechanism: S-Inhibition heads reading S2 through their value branch.

Finally, the same mechanism appears across scales: the S1–S2 credit gap is positive in every Pythia model from 160M to 6.9B, with S2's credit turning outright negative at three of the five scales (Appendix H).

Taken together, the two assumptions appear to suffice: the single algorithm they fix clears an established reference for each readout, with interaction scores predicting knockout effects, rerooted paths recovering causally established compositions branch by branch, and token credit passing perturbation faithfulness. Where the readouts overlap they agree: the suppression mechanism the paths

Table 4: Copying vs. suppression at S2 (GPT-2 small, $n = 100$). Each condition uses a contrastive target (Appendix B): S1→S and S2→S attribute toward $W_U[\mathrm{S}] - W_U[\mathrm{IO}]$, C→C toward $W_U[\mathrm{C}] - W_U[\mathrm{A}]$, and C→B toward $W_U[\mathrm{B}] - W_U[\mathrm{A}]$. Notation and E are defined in the text. For IFR, C→C = C→B, so E is undefined. Entries are percentages of the prompt's total positive credit toward the target; E is unitless.

Method	C→C	S2→S	C→B	S1→S	E
IFR	+7.2	+4.8	+7.2	+11.2	n/a
AttnLRP	+54.5	+2.1	-4.3	+43.7	+0.89
CP-LRP	+73.9	+26.2	+2.5	+51.9	+0.67
UNPACK	+23.9	+7.3	+8.7	+22.2	+1.09

name is visible in the token credit. And where our method diverges from the causal ones, at layer 0, the divergence is itself interpretable as a prompt-independent bias.

5 LIMITATIONS AND CONCLUSION

We asked whether two structural assumptions, linearity of the residual stream and a shared key-value form, suffice for mechanistic interpretability. Our results indicate that they do for all three readouts studied here. A single computational graph provides an established external check for each: interaction scores predict knockout effects, rerooted paths recover causally established compositions branch by branch, and token credit passes a perturbation faithfulness test. The readouts are also connected: token credit decomposes into named paths, and the paths aggregate into component interactions. The systematic divergence from causal methods occurs at layer 0, where it is consistent with a prompt-independent bias. The resulting scores therefore have a definite meaning because they follow from stated assumptions that a reader can accept or reject, rather than from task-specific propagation rules.

Several choices and limitations remain. First, the decomposition is marginal rather than exact through LayerNorm: each component is centered, scaled by the LayerNorm weight, and divided by the full stream's standard deviation, which we treat as constant so that the decomposition remains additive. This treatment is motivated by the pre-norm architecture used by GPT-2 and Pythia; post-norm models require a different treatment and are not tested. The safe denominator and output-aligned projection are additional algorithmic choices, although the results are stable across the configurations examined in Appendix I. Second, the method measures contribution under the computation the model actually performed, not causal effect under intervention. A large contribution need not imply that a component is causally necessary. Our layer-0 results illustrate this distinction: these components receive high credit across prompts, while the causally established components lose credit when the task is removed. Separating bias from signal required a matched counterfactual template, however, and the same diagnosis is unavailable when no such counterfactual exists, as in the greater-than task (Appendix F). Contribution therefore complements intervention rather than replacing it. Whether it can predict interventional effect more generally, or identify a circuit without a counterfactual, remains open.

Finally, the method has computational limits. Its preparation cost is $O(L^2Hn)$ key-side scores for L layers, H heads, and sequence length n. Streaming one source layer at a time keeps memory linear rather than quadratic in L, but preparation time grows with model width and depth. Named-path enumeration is combinatorial in depth, so the pruning threshold τ and the top-k paths retained per rerooting can miss low-magnitude routes in deeper models. The aggregate token attributions are computed exactly; the threshold affects only path enumeration (Appendix A.3).

This view places interaction, composition, and token attribution in one framework. Instead of developing a separate analysis for each question, we can study these aspects of the model through the same graph. This also gives us a clear direction for future work: use the graph to identify important computational routes, test them with interventions, and compare how they change across prompts and model scales. The graph could therefore serve as a common starting point for building and testing circuits.

Reproducibility Statement

Code, prompt sets, and scripts to regenerate all tables and figures are provided as anonymized supplementary material. Algorithm pseudocode and all hyperparameters are in Appendix A; experimental setups are in Appendices C–G.

AI Use Statement

Generative AI tools were used to polish wording and improve early drafts of the text, and to assist with writing code. All text was reviewed and edited by the authors. The main library was verified through unit tests and manual review; the authors take full responsibility for all content and results.

References

Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4190–4197, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.385. URL https://aclanthology.org/2020.acl-main.385/.

Reduan Achtibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. AttnLRP: Attention-aware layer-wise relevance propagation for transformers. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 135–168. PMLR, 21–27 Jul 2024.

Ameen Ali, Thomas Schnake, Oliver Eberle, Grégoire Montavon, Klaus-Robert Müller, and Lior Wolf. Xai for transformers: Better explanations through conservative propagation, 2022. URL https://arxiv.org/abs/2202.07304.

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L Turner, Brian Chen, Craig Citro, et al. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 07 2015. doi: 10.1371/journal.pone.0130140. URL https://doi.org/10.1371/journal.pone.0130140.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. *Proceedings of ICML*, 2023.

Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 782–791, June 2021.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 16318–16352. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Paper-Conference.pdf.

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

Javier Ferrando and Elena Voita. Information flow routes: Automatically interpreting language models at scale. In *Proceedings of EMNLP 2024*, pp. 17432–17445, Miami, FL, USA, 2024.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *Proceedings of EMNLP*, 2021.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 2023.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms, 2024. URL https://arxiv.org/abs/2403.17806.

nostalgebraist. interpreting GPT: the logit lens. *LessWrong*, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 3319–3328. JMLR.org, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. In *ICLR*, 2023.

A Algorithm details

Section 3.2 states the two stages of the recursion. This appendix gives the SafeDenom floor they normalize with (Section A.1), elaborates both stages (Section A.2), and assembles them as pseudocode (Section A.3).

A.1 The SafeDenom soft floor

The recursion repeatedly normalizes a parent's importance by a denominator built from signed child contributions $\{r_i\}$:

$$\text{share}_i = \frac{r_i}{\sum_j r_j}.$$

When positive and negative contributions nearly cancel, $|\sum_j r_j| \ll \sum_j |r_j|$, the per-share factor blows up and amplifies the parent's credit. Over deep recursion this compounds into unstable attributions where a handful of paths receive scores far larger than the model's actual logit can justify.

We control this with a soft floor on the denominator's magnitude:

$$\textsc{SafeDenom}(\{r_i\};\beta) := \operatorname{sign}\Big(\sum_i r_i\Big) \cdot \max\Big(\big|\sum_i r_i\big|,\ \beta \cdot \sum_i |r_i|\Big). \tag{8}$$

The maximum caps the per-step amplification at $1/\beta$. If the signed sum is large in magnitude relative to the absolute mass ($|\sum_i r_i| \geq \beta \sum_i |r_i|$), SafeDenom agrees with the signed sum and recovers the standard normalization. If the signed sum is small (heavy cancellation), the denominator's magnitude is held above β times the absolute mass while its sign is preserved.

We use $\beta = 0.8$ throughout; sensitivity to β is reported in Appendix I.

A.2 THE TWO STAGES IN FULL

Stage 1: credit to the entries. Stage 1 is Eq. 5, instantiated per sublayer. For a head, the entries are source positions s, with $\phi(S)_s = \alpha_{h,q,s}$ and $U_s = X_s W_V W_O$; for an MLP, they are neurons j, with $\phi(S)_j = \phi(\text{pre}_j)$ and $U_j = W_{\text{down}}[j,:]$. All vectors carry the marginal LayerNorm normalization of Section 5.

At the first call $d = d_t$. Deeper in the recursion, d is the output the sublayer actually wrote at q:

$$o_{\text{mlp}}(q) = \sum_j \phi(\text{pre}_j)\, W_{\text{down}}[j,:], \qquad o_{\text{attn}}(q) = \sum_{h' \in \text{layer}} \sum_s \alpha_{h',q,s}\, U_s^{(h')}. \tag{9}$$

$o_{\text{attn}}(q)$ is what the attention sublayer wrote at q, so all heads of a layer are scored against it; the recursion expands one head at a time and renormalizes that head's row with SAFEDENOM. Appendix J ablates this choice.

Stage 2 at a head: the three branches. Each source position's share I_s is split three ways, $M \in \{K, Q, V\}$, with weights $w_K = w_Q = w_V = \frac{1}{3}$. The three have the same form,

$$I_M(c_j) = w_M\, I_s \cdot \frac{s_j^M}{\text{SAFEDENOM}(\{s_m^M\}_m,\, \beta)}, \tag{10}$$

and differ only in the score s_j^M they divide and the position the branch continues at:

M	score s_j^M	upstream components	continues at
K	$\langle X_q,\ c_{js} \rangle_{\text{QK}}$	writers at s	s
Q	$\langle c_{jq},\ X_s \rangle_{\text{QK}}$	writers at q	q
V	$\langle c_{js} W_V,\ X_s W_V \rangle$	writers at s	s

K and Q are the key-side and query-side splits of Section 3.1, the same bilinear form read on its two arguments. V is a component's share of the value the head actually formed at s, so the V scores sum to $\|X_s W_V\|^2$; it is taken in head space, before W_O, which is applied to the head's output as a whole and scales every component's share alike. No credit direction enters any of the three: each divides I_s by the upstream components' shares of a quantity the forward pass already computed.

Stage 2 at an MLP: one branch. The values $W_{\text{down}}[j,:]$ are parameters, so only the score carries upstream credit. Neuron j's share is distributed over the components c_{kq} writing the residual at q, by their contribution to its pre-activation, $s_k^{\text{mlp}}(q)[j] = c_{kq} W_{\text{up}}[:, j]$, normalized by SAFEDENOM as above. Neurons are treated independently, and credit flowing through different neurons recombines at each upstream component. The recursion continues at q.

A.3 PSEUDOCODE

Algorithm 1 assembles the two stages. Each component is passed down with the position it sits at: the K- and V-branches continue at the source position, the Q-branch at the query position.

Here $o_{\text{attn}}(q)$ and $o_{\text{mlp}}(q)$ are the outputs the head and the MLP actually wrote at q, the directions of Eq. 9. They are passed down only so that the next call's Stage 1 has a direction to project on; Stage 2 never uses one.

The pruning threshold τ keeps the recursion tree tractable by dropping sub-credits below τ in absolute value, which is necessary because the tree expands by roughly $K \cdot S$ per level (components times sequence length). The aggregate token-level credit reported in Section 4 is computed by an equivalent vectorized procedure that does not depend on τ and so gives the exact attribution; only named-path enumeration uses $\tau = 10^{-3}$.

B EDGE CASES

This appendix records small implementation choices that affect the numbers reported in the paper but are not part of the main algorithm description.

Algorithm 1 Recursive backward attribution

Require: target token t, position p, pruning threshold τ, amplification floor β (Eq. 8)
$d_t \leftarrow$ target direction (Eq. 4)
$I(c_k) \leftarrow \langle c_k(p),\, d_t \rangle$ for each residual component c_k at position p
RECURSE($\{(c_k,\, p,\, I(c_k))\}_k,\, d_t$)

Procedure RECURSE(items, d)
for each (component c, position q, credit I) in items with $|I| \geq \tau$ **do**
  if c is the embedding **then**
    $\mathrm{credit}[q]$ += I
  else if c is attention head (L, h) **then**
    $\{(s, I_s)\} \leftarrow$ STAGE1(I, d), entries = source positions (Eq. 5)
    for each s with $|I_s| \geq \tau$ **do**
      RECURSE$(\{(c_j, s, I_K)\}_j \cup \{(c_j, q, I_Q)\}_j \cup \{(c_j, s, I_V)\}_j,\;\; o_{\text{attn}}(q))$ (Eq. 10)
    end for
  else if c is MLP at layer L **then**
    $\{(j, I_j)\} \leftarrow$ STAGE1(I, d), entries = neurons (Eq. 5)
    for each neuron j with $|I_j| \geq \tau$ **do**
      RECURSE$(\{(c_k,\, q,\, I_j(c_k))\}_k,\;\; o_{\text{mlp}}(q))$
    end for
  end if
end for

Bias terms. Attention and MLP bias parameters are constants and so carry no per-input information. We drop them from the residual decomposition; the recursion does not visit bias components.

Logit-difference targets. For contrastive tasks such as IOI, we replace $W_U[t] - \overline{W}_U$ in Eq. 4 with $W_U[t] - W_U[t']$, where t' is the distractor token (e.g. the subject S in IOI). Then $\langle c_k, d_t \rangle$ is what c_k contributes to the logit *difference* $z_t - z_{t'}$ rather than to z_t alone. Vocabulary-mean centering is unnecessary in this case, since a constant added to every logit cancels in the difference.

Pre-norm assumption. The marginal-LayerNorm normalization of Section 5 treats each component c_k as passing through one LayerNorm before being projected, which matches the pre-norm architecture used by GPT-2 and the Pythia family. Post-norm models (e.g. the original BERT) place LayerNorm after the residual addition rather than before projection, which would require a different decomposition.

C KNOCKOUT VALIDATION

Section 4.1 states the setup and the Pythia-6.9B result; this appendix gives the scoring formulas, the sampling, and the full five-model correlations. All models are taken at training step 143,000.

Scores and ablation. The *attention strength* of a component k is its interaction score of Eq. 6 summed over all downstream heads m and query positions q, $\sum_{m,q} \text{score}^{\text{attn}}_{k \to m}$; the *MLP strength* is the analogous sum of Eq. 7 over downstream MLPs, $\sum_{m,q} \text{score}^{\text{mlp}}_{k \to m}$. The Δppl of the ablation of Section 4.1 is measured on 200 held-out Pile sentences (7,967 tokens). For tractability we select, per source layer, 5 components spread across the range of observed scores, yielding 56, 116, 116, 156, and 156 knockouts at the 160M, 410M, 1.4B, 2.8B, and 6.9B scales respectively.

Results. Figure 4 plots score against knockout effect for each of the five models; the monotonic relationship of Figure 2 holds on both channels at every scale. Table 5 reports Spearman rank correlations: within-layer mean ρ between 0.86 and 0.96 on the MLP channel and between 0.72 and 0.86 on the attention channel. The MLP channel is cleanly ranked at every scale; the attention channel is noisier but still strong, reflecting that attention strength predicts a more indirect outcome (a perturbation to downstream attention patterns) than MLP strength does (a direct perturbation to downstream MLP activations). Two qualitative patterns appear in every model: the embedding

(layer -1) is a high-end outlier on the MLP-cut channel, with $\Delta\text{ppl} > 10^3$ at every scale; and layer identity is visible as a colour gradient from purple (early) to yellow (late), with early layers dominating the MLP channel and late-layer attention heads dominating the attention channel.

Table 5: Spearman rank correlation between interaction score and knockout Δppl. *Within-layer* ρ is the mean of per-source-layer Spearman correlations (the stricter test, holding the set of downstream receivers fixed). *Cross-layer* ρ pools all components for a given channel.

		within-layer ρ		cross-layer ρ	
Model	n	attn	mlp	attn	mlp
Pythia-160M	56	0.72	0.96	0.84	0.95
Pythia-410M	116	0.82	0.91	0.88	0.87
Pythia-1.4B	116	0.73	0.86	0.78	0.82
Pythia-2.8B	156	0.73	0.87	0.67	0.86
Pythia-6.9B	156	0.86	0.90	0.76	0.87

D COMPOSITION RANKING MATRICES

Table 6 reports composition rankings for the first upstream head from each role tier, using attention heads from layer 1 onward. Each cell gives the median rank and percentage of rerootings in which the upstream role was found; lower rank indicates stronger composition. Figure 5 illustrates the route-level view for the top-3 V-mode paths from a rerooting at S-Inhibition head A8.H6.

Table 6: Composition ranking matrices (n=100, GPT-2 small). Each cell: median rank / found%. Rows = head of interest, columns = upstream role. Lower rank indicates stronger composition.

Filter	↓ / ↑	NM	S-Inh	Ind	Dup	Prev
All modes	NM	6 / 33%	4 / 100%	14 / 84%	22 / 29%	21 / 77%
	S-Inh	–	20 / 32%	4 / 96%	2 / 97%	20 / 52%
	Ind	–	–	3 / 25%	2 / 100%	4 / 98%
K-mode	NM	–	10 / 48%	14 / 27%	13 / 19%	14 / 46%
	S-Inh	–	–	5 / 93%	4 / 91%	13 / 4%
	Ind	–	–	21 / 3%	8 / 10%	3 / 98%
Q-mode	NM	2 / 33%	1 / 100%	4 / 63%	18 / 5%	10 / 56%
	S-Inh	–	9 / 32%	3 / 66%	13 / 35%	11 / 42%
	Ind	–	–	2 / 25%	1 / 100%	10 / 36%
V-mode	NM	–	–	4 / 24%	6 / 10%	6 / 9%
	S-Inh	–	–	2 / 96%	1 / 95%	10 / 21%
	Ind	–	–	8 / 10%	11 / 32%	9 / 88%

Table 7 repeats the rankings with layer-0 heads included. This mainly introduces two Duplicate Token heads, `attn_0_head_1` and `attn_0_head_10`, which move the Dup column to ranks 1–2 with near-100% coverage. Because these heads write at every position, the same pattern also appears on the ABC counterfactual and therefore does not provide additional evidence for task-specific Dup composition. Other columns retain essentially the same coverage while their median ranks increase by about 10 as the pool gains competitors.

E COVERAGE-DROP DETAILS

Table 8 gives a per-head view of the coverage drop, complementary to Figure 3: the figure counts appearances through the claimed branch only, on the prompts where the model fails the ABC control; the table sums credit over the K, Q and V branches on all 100 prompt pairs. For each hypothesis of Table 2 it lists the five strongest layer-$\geq$1 and five strongest layer-0 heads on a single ranking over all attention heads, re-rootings paired across arms. Under both views the predicted tiers lose share

Filter	↓ / ↑	NM	S-Inh	Ind	Dup	Prev
All modes	NM	18 / 33%	15 / 100%	26 / 84%	2 / 100%	33 / 77%
	S-Inh	–	32 / 32%	12 / 96%	1 / 100%	31 / 52%
	Ind	–	–	13 / 25%	1 / 100%	12 / 98%
K-mode	NM	–	21 / 48%	25 / 27%	5 / 100%	24 / 46%
	S-Inh	–	–	13 / 93%	2 / 100%	23 / 4%
	Ind	–	–	32 / 3%	2 / 100%	7 / 98%
Q-mode	NM	6 / 33%	3 / 100%	9 / 63%	6 / 89%	18 / 56%
	S-Inh	–	18 / 32%	10 / 66%	3 / 100%	19 / 42%
	Ind	–	–	5 / 25%	1 / 100%	20 / 36%
V-mode	NM	–	–	14 / 24%	1 / 100%	16 / 9%
	S-Inh	–	–	5 / 96%	1 / 100%	17 / 21%
	Ind	–	–	19 / 10%	2 / 100%	18 / 88%

Table 7: Composition rankings with layer-0 attention heads included. Compare Table 6 for the layer-$\geq$1 pool.

and coverage on the counterfactual while layer-0 heads hold near-100% coverage on both arms, the bias behavior of Appendix D.

F Composition on the greater-than circuit

Setup. Prompts and target follow the released code of Hanna et al. (2023). The template is "`The {noun} lasted from the year XXYY to the year XX`", with their 120 FrameNet nouns and the 768 two-token years in $[1000, 1900)$. Single-token years and each century's first and last valid year are excluded, so every prompt has at least one valid and one invalid two-digit answer; YY is swept balanced over the cutoff, and we do not filter for whether GPT-2 answers correctly. Their metric is a group quantity, the probability difference $\sum_{y>\mathrm{YY}} p_y - \sum_{y\leq \mathrm{YY}} p_y$, so we attribute toward the difference of the mean unembedding rows of the valid and the invalid year sets, a logit-difference target in the sense of Appendix B.

Claims and procedure. Their §3.2 builds the circuit by iterated path patching and states a set of composition claims, each naming what a given receiver relies on. We check the four whose named sources sit above layer 0, since a layer-0 component contributes strongly to every later layer and ranks high whatever it feeds:

1. MLP 9 relies on a9.h1.
2. MLP 8 relies on a8.h11, a8.h8, a7.h10, a6.h9, a5.h5 and a5.h1.
3. MLPs 8–11 rely on the MLPs upstream of them.
4. Those seven heads read their values, at the YY position, from MLPs 1–3.

For each claim we reroot at the receivers it names, at the end position for claims 1 to 3 and at the YY position for claim 4, and rank the components feeding each receiver by the credit they send in. A claim names a group of components rather than a single one, so we score it by the best-placed member of the group, as Section 4.2 does for the role tiers on IOI; ranks are medians over $n = 100$ prompts. Each rank is reported twice, against all upstream components and with the layer-0 heads, $\mathrm{mlp_0}$ and the embedding taken out of the pool; without the second, layer-0 components fill the top of nearly every ranking and there is little to compare.

All four claims place the group they name in the top four once layer-0 writers are out of the pool (Table 9), against pools of 17 to 60 components. Claim 3 is the cleanest: $\mathrm{mlp_8}$ is the single largest contributor to $\mathrm{mlp_9}$ in either pool. Against all sources the same claims sit near rank 10, and for most of them that gap is layer-0 crowding rather than a weaker edge: everything ranked above the group into $\mathrm{mlp_8}$, and above the value inputs of a9.h1 and a6.h9, is an embedding or a layer-0 writer. a5.h1 is the exception, with layer-1 and layer-2 heads above its value inputs as well.

Table 8: Per-head coverage and credit share for the three composition hypotheses (n=100, GPT-2 small). The five strongest heads are shown for each layer split. IOI and ABC rerootings are paired and use a single ranking over all heads. Share is mean credit fraction, with absences counted as zero; Cov. is the fraction of rerootings in which the head appears; r_{ABC} is its ABC rank. Layer-0 heads retain near-100% coverage on both arms and therefore do not distinguish the task from its counterfactual.

		Credit share %			Coverage %			
Head	Tier	IOI	ABC	rel.	IOI	ABC	Δ	r_{ABC}
H1 NM ← S-Inh	*root: name movers, END; predicts S-Inh*							
layer≥1								
attn_1_head_4	–	2.26	1.53	-32%	94.3	83.3	-11.0	14
attn_1_head_8	–	2.24	1.94	-13%	90.7	95.7	+5.0	13
attn_1_head_1	–	1.90	1.17	-38%	96.3	92.0	-4.3	15
attn_8_head_6	S-Inh	1.46	0.60	-59%	100.0	91.3	-8.7	17
attn_1_head_10	–	1.40	1.06	-24%	100.0	98.0	-2.0	16
layer 0								
attn_0_head_8	–	13.26	13.61	+3%	100.0	100.0	+0.0	2
attn_0_head_1	Dup	13.09	16.71	+28%	100.0	100.0	+0.0	1
attn_0_head_9	–	9.77	9.11	-7%	100.0	100.0	+0.0	5
attn_0_head_4	–	9.32	10.73	+15%	100.0	100.0	+0.0	4
attn_0_head_5	–	9.25	11.32	+22%	100.0	100.0	+0.0	3
H2 S-Inh ← Dup/Ind	*root: S-Inh heads, END; predicts Dup/Ind*							
layer≥1								
attn_1_head_8	–	6.15	7.71	+25%	99.5	100.0	+0.5	6
attn_3_head_0	Dup	4.39	0.21	-95%	96.5	55.5	-41.0	27
attn_1_head_4	–	3.34	4.87	+46%	100.0	100.0	+0.0	8
attn_5_head_5	Ind	3.01	0.25	-92%	95.0	57.2	-37.8	25
attn_1_head_1	–	1.25	1.41	+12%	92.2	94.0	+1.8	14
layer 0								
attn_0_head_11	–	13.86	14.40	+4%	100.0	100.0	+0.0	2
attn_0_head_10	Dup	12.09	9.08	-25%	100.0	100.0	+0.0	4
attn_0_head_1	Dup	10.82	14.65	+35%	100.0	100.0	+0.0	1
attn_0_head_3	–	8.80	10.44	+19%	100.0	100.0	+0.0	3
attn_0_head_4	–	7.60	9.00	+18%	100.0	100.0	+0.0	5
H3 Ind ← Prev	*root: induction heads, S2; predicts Prev/Dup*							
layer≥1								
attn_3_head_0	Dup	5.58	0.47	-92%	100.0	61.3	-38.7	32
attn_1_head_8	–	3.33	2.97	-11%	98.8	99.8	+1.0	12
attn_1_head_9	–	3.29	6.47	+97%	99.8	100.0	+0.2	4
attn_1_head_4	–	3.25	7.40	+127%	100.0	100.0	+0.0	3
attn_4_head_11	Prev	2.92	0.46	-84%	97.2	80.5	-16.8	33
layer 0								
attn_0_head_1	Dup	13.12	10.27	-22%	100.0	100.0	+0.0	1
attn_0_head_9	–	8.30	10.00	+21%	100.0	100.0	+0.0	2
attn_0_head_11	–	6.59	5.26	-20%	100.0	100.0	+0.0	8
attn_0_head_10	Dup	5.50	5.22	-5%	100.0	100.0	+0.0	9
attn_0_head_5	–	5.25	6.17	+17%	100.0	100.0	+0.0	5

Table 10 gives the per-source detail behind the group ranks. The spread inside a group is wide: of the six heads feeding $\mathrm{mlp_8}$, a5.h5 reaches rank 2 while a8.h8 sits at 31. Claim 3 holds through the adjacent hops and weakens with distance, from rank 1 for $\mathrm{mlp_8} \to \mathrm{mlp_9}$ to 27 for $\mathrm{mlp_8} \to \mathrm{mlp_11}$. On claim 4 every head but a5.h1 finds one of MLPs 1–3 in its top three, matching the joint form of the claim.

Circuit components outrank the rest beyond the named connections (Table 11): median rank 10 against 22 over all sources, and 8 against 14 at layer ≥ 1. That the gap survives removing layer 0 rules out the reading that circuit components rank high only because early writers do and the circuit

Table 9: Composition claims 1–4 of Hanna et al. (2023) ($n = 100$, GPT-2 small). Each entry is the median rank of the best-placed member of the group they name, in the receiver's upstream ranking, against all upstream components and with layer-0 writers removed. *recv.* is how many receivers the claim covers; *pool* is how many components the ranking runs over, so rank 2 of 17 and rank 2 of 60 can be told apart.

			median rank		pool size	
#	claim	recv.	all	layer ≥ 1	all	layer ≥ 1
1	MLP 9 relies on a9.h1	1	9.5	3.5	69	55
2	MLP 8 relies on six heads at layers 5–8	1	10	2	74	60
3	MLPs 8–11 rely on the MLPs upstream	3	4	2	60–69	47–55
4	the seven heads read values from MLPs 1–3	7	9	2	31–39	17–23

Table 10: Per-source detail for claims 2, 3 and 4, as *all / layer* ≥ 1. A group rank reports its best member; the spread behind it is wide, and two named sources ($\mathrm{a8.h8} \to \mathrm{mlp_8}$, $\mathrm{mlp_8} \to \mathrm{mlp_11}$) stay low in both pools.

claim 2: into mlp_8		claim 3: MLP to MLP		claim 4: best of MLPs 1–3	
source	rank	edge	rank	receiver	rank
a5.h5	10 / 2	$8 \to 9$	1 / 1	a9.h1	5 / 1
a7.h10	16 / 7	$8 \to 10$	4 / 2	a7.h10	6 / 2
a6.h9	21 / 10	$9 \to 10$	5 / 2	a8.h11	8 / 2
a8.h11	29 / 17	$10 \to 11$	10 / 3	a8.h8	9 / 3
a5.h1	29 / 18	$9 \to 11$	15 / 4	a5.h5	9.5 / 3
a8.h8	44 / 31	$8 \to 11$	39 / 27	a6.h9	10 / 2
				a5.h1	26 / 14

contains early writers. Which of these contribution edges are causally necessary is a counterfactual question; Section 5 discusses the gap between contribution and causal effect.

G Setup for the token-attribution checks

Sentences. The 300 Pile sentences are 8 to 45 tokens long, kept when the model's argmax next token is a confident content word ($p \geq 0.1$, alphabetic). Without this filter, two thirds of the targets are newline and punctuation defaults, whose likelihood increases as context is destroyed. Token positions are preserved during flipping, so the sequence length is constant, and BOS is excluded from ranking and flipping as a causally inert attention sink.

Readout. We record $\mathrm{softmax}(z)_t$ at each flipping step rather than the raw logit z_t used by the original protocol. A softmax model's logits are identified only up to an additive constant: adding c to every logit leaves the predicted distribution unchanged. A change in z_t is therefore not by itself a change in the model's behaviour, since it may consist largely of a vocabulary-wide shift that the softmax cancels, and it can move in the opposite direction to the probability the model assigns to the target. The probability is what the protocol means to measure, and it is invariant to the shift by construction.

Methods. **IFR** (Information Flow Routes) is re-implemented from its definition, since the released code is licensed CC BY-NC; its proximity score is $\max(\|y\|_1 - \|z - y\|_1, 0)$, non-negative by construction and target-agnostic. **AttnLRP** and **CP-LRP** use the authors' LXT library. Each method runs in its own process, since LXT patches the model globally.

Target direction. Each method attributes backward from a single number at the output, a scalar readout of the target token, and we give every method the same one. The choice matters. LRP relevance is conservative, so the credit it assigns sums to exactly that number; taking GPT-2's raw target logit (around -100) therefore makes the relevance explain that arbitrary offset and inverts

Table 11: Circuit components against everything else, ranked in the same pool. Median rank over every component appearing in any rerooting, split by whether Hanna et al. (2023) name it anywhere in the circuit. Unlike the per-claim ranks above, this comparison is not fixed by construction, since both groups are ranked against the same competitors.

pool	circuit	non-circuit
all sources	10 ($n = 246$)	22 ($n = 661$)
layer ≥ 1	8 ($n = 121$)	14 ($n = 498$)

the heatmap: the relevance sum is negative on 99.5% of examples and AttnLRP scores 6% faithful. We instead use the vocabulary-mean-centered logit $z_t - \bar{z}$, or a logit difference where a contrastive target exists (IOI; Appendix B). LXT's own GPT-2 module flags this issue ("GPT2 has negative logit values ... we must also explain the softmax to kick out the negative bias").

H S2 SUPPRESSION ACROSS THE PYTHIA-DEDUPED FAMILY

Section 4.3 defines the four credit quantities and the suppression index E and reports them on GPT-2 small. Table 12 reports the same S2 comparison across five Pythia-deduped scales.

Table 12: S2 suppression across Pythia-deduped scales ($n = 100$, contrastive targets as in Table 4).

Model	C→C	S2→S	C→B	S1→S	S1–S2 gap
160M	+30.9	+1.5	+14.1	+26.2	+24.7
410M	+19.9	−18.1	+1.9	+48.8	+66.9
1.4B	+15.5	−39.6	+4.4	+41.3	+80.9
2.8B	+32.5	+3.4	+5.1	+35.1	+31.7
6.9B	+14.7	−29.3	+4.1	+33.9	+63.2

The S1–S2 gap is positive at every scale. S2 credit is near zero or negative at every scale except 2.8B, where the gap narrows and S2 retains moderate positive credit, suggesting a less structured IOI circuit at this scale.

I SENSITIVITY TO β

The SAFEDENOM floor β (Eq. 8) caps per-step amplification when positive and negative contributions nearly cancel. The knockout scores of Section 4.1 are read off the key-side decomposition directly (Eqs. 6, 7), before the recursion runs, so they are β-independent by construction; the paths and the token attribution pass through SAFEDENOM at every step. We sweep β against the perturbation-faithfulness metric of Section 4.3 (Table 13), using the same probability readout and area statistic on 15 short Pile prompts.

The floor matters only in that it must be on. With $\beta = 0$, a near-cancelling denominator lets a single step inflate credit arbitrarily and the ranking degrades to chance ($\%f = 53$). Once $\beta \geq 0.2$ the amplification is capped and faithfulness is stable, with no trend across the range and $\beta = 0.8$ (our default throughout) indistinguishable from its neighbours.

J ABLATION: OUTPUT-ALIGNED DISPATCH

One choice in the recursion is not forced by the structure. Beyond the first step, credit is split across a component's entries by projecting each entry's gated value onto the direction the component actually wrote (Eq. 5 with $d = \phi(S)U$). The alternative is to split by the gate alone, normalized: the attention weight $\alpha_{h,q,s}$ for a head, the activation $\phi(\text{pre}_j)$ for an MLP. The gate already says how strongly each entry was selected, so why also ask where the component wrote? This appendix answers that. Everything else about the decomposition is held fixed.

Table 13: Perturbation faithfulness across β (15 Pile prompts, probability readout; ΔA and %f as in Table 3). At $\beta = 0$ the floor is off, per-step credit amplifies without bound, and faithfulness collapses to chance. For every $\beta \geq 0.2$ it is flat, with $\beta = 0.8$ in the middle of the plateau.

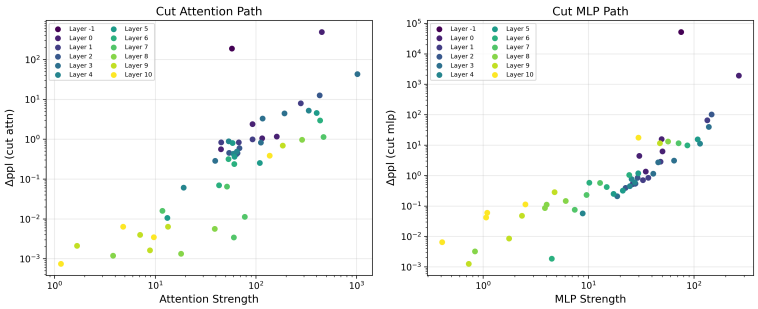
β	ΔA	%f
0.0	$+0.002$	53
0.2	$+0.046$	100
0.4	$+0.045$	100
0.6	$+0.060$	100
0.8	$+0.046$	100
1.0	$+0.053$	100
1.5	$+0.045$	100
2.0	$+0.045$	100

The attention sink. BOS is an attention sink. It receives large attention weight, but its value contributes almost nothing to what a head writes into the residual stream. Gate-alone dispatch splits credit by that attention weight, so it hands position 0 a large share. Projecting onto the realized output (Eq. 5) instead weights each entry by what it actually wrote, so the sink falls away on its own, with no BOS filtering anywhere in the pipeline.

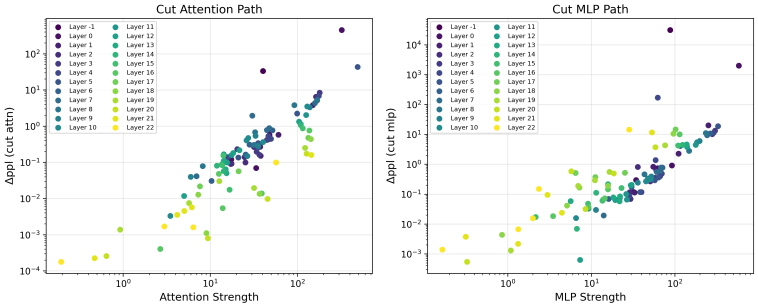
We measure the effect on 50 prompts each from three sources (Table 14): Pile sentences, factual-recall prompts, and IOI. Gate-alone inflates the BOS credit share on all three, but how much that matters depends on the input. On short, low-content prompts (IOI, factual recall) the sink can outrank the real tokens, taking top-1 on 32% and 18% of the prompts, while on general text enough later content outweighs it and it wins only 4% of the time. Output-aligned dispatch never lets the sink win, on any source.

Table 14: The BOS attention sink under the two dispatch rules (50 prompts each, GPT-2 small). *Share*: mean percentage of positive token-attribution credit on position 0. *Top-1*: fraction of prompts where position 0 is the single highest-credited position.

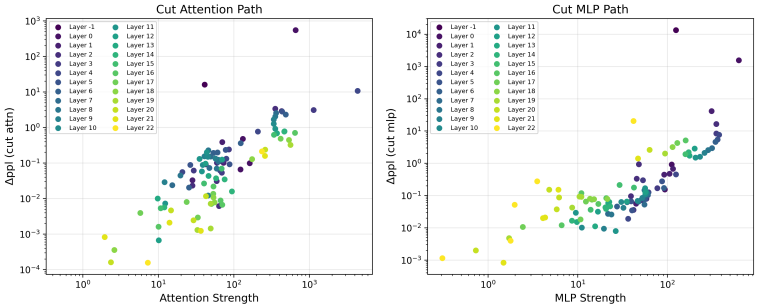
	Pile		Factual		IOI	
Dispatch	share	top-1	share	top-1	share	top-1
gate alone (activation-weighted)	9.2%	4%	22.3%	18%	28.9%	32%
output-aligned (ours)	6.5%	0%	13.0%	0%	18.8%	0%



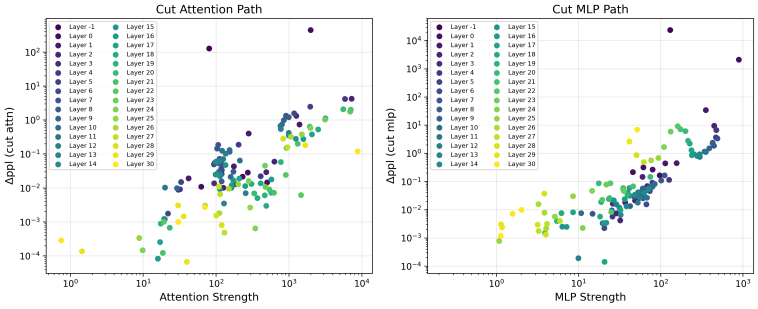
(a) Pythia-160M-deduped (12 layers, 56 knockouts).



(b) Pythia-410M-deduped (24 layers, 116 knockouts).



(c) Pythia-1.4B-deduped (24 layers, 116 knockouts).



(d) Pythia-2.8B-deduped (32 layers, 156 knockouts).

Figure 4: Knockout validation across the Pythia-deduped family; Pythia-6.9B is shown in Figure 2. Each point is one residual-stream component: the x-axis its aggregated interaction score, the y-axis Δppl when the channel is ablated (left panels: attention; right panels: MLP). Points are colored by source layer (darker $=$ earlier; layer -1 is the embedding); both axes log-scaled. The monotonic relationship is stable across the $40\times$ parameter range.

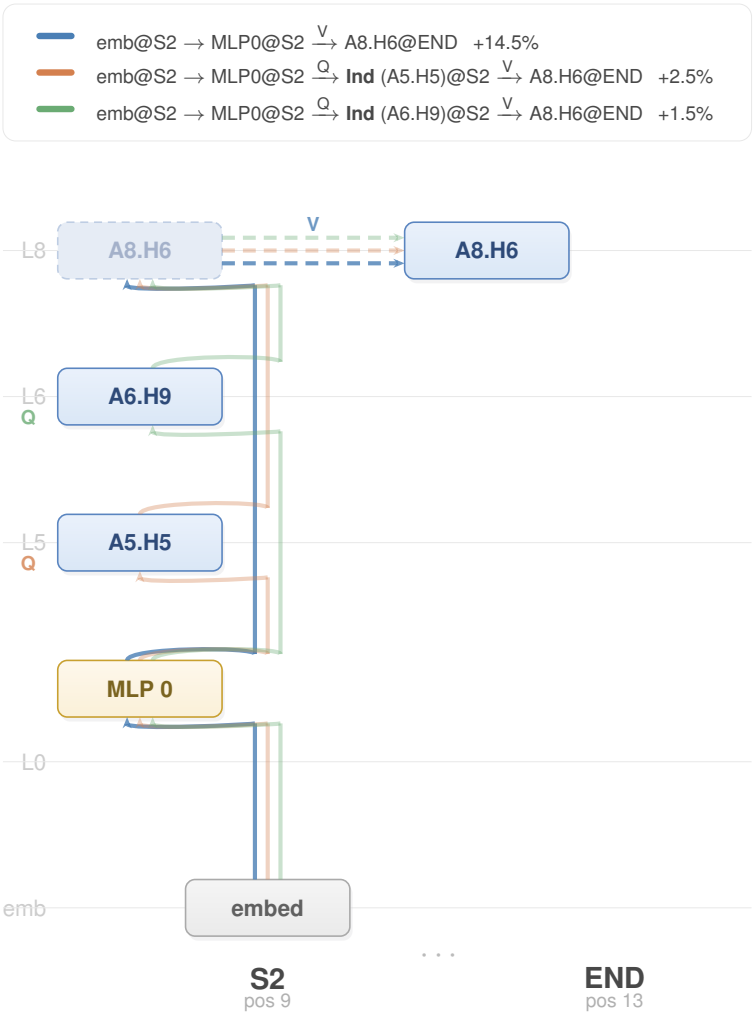


Figure 5: Top-3 V-mode composition paths from a rerooting at S-Inhibition head A8.H6. The paths trace upstream computation through cross-position attention; the second and third paths pass through Induction heads A5.H5 and A6.H9, illustrating the Ind→S-Inh composition. Percentages are shares of the total positive credit entering the root `A8.H6@END`.