

Universiteit Leiden

Master Computer Science

Every Component is a Lookup: Token Attribution and Composition from a Single Decomposition

Name:	Po-Kai Chen
Student ID:	s4283341
Date:	19/06/2026
Specialisation:	Artificial Intelligence
1st supervisor:	Niki van Stein
2nd supervisor:	Aske Plaat

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Mechanistic interpretability of transformers requires identifying not just which components matter but how they compose into the computational route that produced a prediction. Both attention and MLP follow a shared key-value template $\phi(S)U$. We exploit this structure to develop UNPACK, a backward recursion that decomposes credit through both sublayers, producing interaction strengths between any two components, named end-to-end paths with K/Q/V composition labels, and per-token attribution from a single forward pass, without intervention, gradients, or auxiliary training. We evaluate on the indirect object identification task. On GPT-2 small, the method recovers all three composition connections described by Wang et al. [2023], including the mode-specific routing of each connection (K, Q, or V). To test token-level attribution beyond trivial copying, we compare two occurrences of the same name in the same decomposition: the first mention retains strong credit while the duplicate-detection position is suppressed, a pattern absent in matched control prompts. Across the Pythia family from 160M to 6.9B parameters, this suppression pattern is consistently recovered at every scale, demonstrating that the method tracks mechanistic structure without ground-truth circuit labels. Code is available at `https://github.com/Fun-Cry/unpacklm`.

1 Introduction

Mechanistic interpretability of transformers requires more than identifying *which* components matter for a given prediction; it requires identifying *how* those components compose into the computational route that produced the prediction. Existing approaches to this problem split into three rough camps. Interventional methods [Wang et al., 2023, Conmy et al., 2023, Syed et al., 2024, Nanda, 2023] measure a component's effect by patching it with a counterfactual activation; the recovered circuits are causally grounded but cost on the order of one forward pass per edge or per gradient step. Replacement-model methods [Ameisen et al., 2025] train auxiliary networks whose sparse features serve as the building blocks of an attribution graph; this gives rich per-feature circuits but requires per-model training. Direct-attribution methods such as DLA [nostalgebraist, 2020] and information flow routes [Ferrando and Voita, 2024] are cheap and intervention-free, but DLA captures only direct contributions to the target logit and information flow routes are unsigned and decompose only the value pathway.

This paper develops a non-interventional attribution method that does not have these gaps. The starting point is a structural observation: both sublayers of a transformer block read information through the same key-value pattern. Self-attention computes scores $QK^\top/\sqrt{d_{\text{head}}}$, applies softmax, and uses the result to read values. Geva et al. [2021] pointed out that an MLP can be written analogously, with the up-projection's columns serving as keys, the activation function as the selection mechanism, and the down-projection's rows as values. Attention and MLP are then both instances of the same template $\phi(S)\,U$, with attention's S a bilinear function of the residual stream and MLP's S a linear one. Once both sublayers are read this way, attribution follows from a single recursion: at each component, decompose the selection score S into per-component contributions from upstream residual writers, propagate credit backward through the keys and values, and continue until credit lands on the input embeddings.

The method produces three levels of output from a single forward pass. First, the scoring functions (Equations 7, 8) quantify the communication strength between any two model components in the network. Second, recursing backward through these components along K, Q, and V composition branches produces an enumerated list of end-to-end paths from input tokens to the target, each carrying a signed magnitude. Third, summing path magnitudes at each input position yields a per-token attribution vector.

We evaluate token-level attribution in Section 4.2 and composition structure in Section 4.3, using the indirect object identification (IOI) task introduced by Wang et al. [2023]. Our contributions are:

- **A unified key-value perspective and corresponding decomposition method.** We show that both attention and MLP can be written in the form $\phi(S)\,U$ and develop a backward decomposition recursion built on this view that produces component-level communication

scores, named end-to-end paths, and per-token attribution from a single forward pass, without intervention, gradients, or auxiliary training. The named paths expose not just which components contribute but through which composition pathways (K, Q, V) they do so, enabling direct inspection of multi-step information flow.

- **Token attribution that distinguishes circuit mechanisms from copying.** On GPT-2 small, the method ranks the indirect object as the highest-attributed token on 97% of IOI prompts. To disentangle this from the trivial copying signal, we design a three-way comparison at the duplicate name position S2, showing that the IOI circuit erases S2's self-promotion credit entirely while the first mention S1 retains it, a pattern absent in matched ABC control prompts.
- **Cross-scale attribution on the Pythia family.** We apply the method to five Pythia-deduped models from 160m to 6.9b parameters. Token attribution ranks IO as the top-credited token on every prompt from 410M onward , and the S1–S2 suppression pattern is consistently recovered across all scales, demonstrating that the method identifies the duplicate-detection mechanism without requiring ground-truth circuit labels.

We call this method UNPACK (**Un**ified **P**ath **A**ttribution through **C**omponent **K**eys) and release it as an open-source library.[1]

2 Background and Related Work

Transformer architecture. A Transformer block [Vaswani et al., 2017] consists of multi-head self-attention and a Multi-Layer Perceptron (MLP). Consider an input tensor $X \in \mathbb{R}^{n \times d_{\text{model}}}$ representing a d_{model}-dimensional vector sequence of length n. Each attention head computes

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{\text{head}}}}\right)V, \tag{1}$$

where Q, K, V are the outputs of X under linear maps $W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$. The intermediate matrix $QK^T \in \mathbb{R}^{n \times n}$ has (i, j)-entry $X_i W_Q W_K^T X_j^T$, a bilinear form $\langle X_i, X_j \rangle_{\text{QK}} := X_i W_Q W_K^T X_j^T$ with X_i as query and X_j as key.

An MLP consists of two linear maps with weights $W_{\text{up}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{mlp}}}$ and $W_{\text{down}} \in \mathbb{R}^{d_{\text{mlp}} \times d_{\text{model}}}$ and a pointwise activation ϕ (biases omitted):

$$\text{MLP}(X) = \phi(XW_{\text{up}})W_{\text{down}}. \tag{2}$$

MLPs are applied either in parallel with [Black et al., 2022, Biderman et al., 2023] or sequentially after multi-head self-attention. In both cases the MLP output and the projected attention output are added back to the residual stream.

Residual stream. Elhage et al. [2021] view the residual stream as the principal communication channel within a Transformer: each block reads from it, computes a function of its current value, and writes back. The residual state at any layer decomposes as $X = \sum_k c_k$, where each c_k is a contribution from a specific component: an attention head output, an MLP output, or the input embedding. This decomposition underlies most post-hoc interpretability of Transformer models, because each c_k has an independent geometric direction in $\mathbb{R}^{d_{\text{model}}}$ that can be examined on its own.

K, Q, V composition. The residual decomposition $X = \sum_k c_k$ becomes useful when paired with the linearity of attention's projection matrices. For an attention head with weights W_Q, W_K, W_V, the query, key, and value vectors decompose by linearity:

$$XW_Q = \sum_k c_k W_Q, \quad XW_K = \sum_k c_k W_K, \quad XW_V = \sum_k c_k W_V, \tag{3}$$

so each upstream component contributes independently to each of the three input pathways. Elhage et al. [2021] call these *Q-, K-, and V-composition*: a head can compose with an earlier head's output through any of its three inputs, and the three composition modes have distinct mechanistic

[1] https://github.com/Fun-Cry/unpacklm

signatures. They study this primarily for head-to-head composition; the same decomposition applies symmetrically to MLPs, whose pre-activation $XW_{\text{up}} = \sum_k c_k W_{\text{up}}$ is linear in the residual state. We make use of this property in Section 3: a backward attribution that traces credit through all three attention pathways and through the MLP pre-activation can recover contributions that one-pathway attribution misses.

The IOI task and its canonical circuit. Indirect object identification (IOI; Wang et al., 2023) requires the model to predict the indirect object in sentences of the form "Alice and Bob went to the store. Bob gave a drink to", where the correct completion is "Alice". Wang et al. [2023] identified a 26-head circuit in GPT-2 small responsible for this task, classifying heads into functional roles: name movers that copy the IO token to the output, S-inhibition heads that attend from the end position to S and suppress it, and duplicate-token heads that detect the repeated name. The key mechanistic insight is that the second mention of S (the S2 position) serves as the trigger for duplicate detection, which in turn activates S-inhibition. A standard control is the ABC variant, where each name is distinct, removing the duplicate signal and deactivating the circuit. We use this well-understood task as a testbed throughout: token attribution experiments (Section 4.2) test whether the method recovers the expected credit pattern across name positions, composition analysis (Section 4.3) tests whether the method identifies known head-to-head interactions, and cross-scale experiments (Section 4.4) test whether these patterns generalize across the Pythia-deduped family [Biderman et al., 2023] from 160 M to 6.9 B parameters.

Attribution and circuit discovery methods. Existing approaches to understanding Transformer internals span several levels of granularity. At the token level, integrated gradients [Sundararajan et al., 2017] and attention rollout [Abnar and Zuidema, 2020] assign importance to input positions but do not decompose the target logit into named computational routes. At the component level, direct logit attribution (DLA; nostalgebraist, 2020) projects each component's output onto the target unembedding direction, capturing direct contributions but missing influence mediated through later components. Information Flow Routes (IFR; Ferrando and Voita, 2024) propagates L_1-mass backward through the residual stream, producing per-token and per-component scores, but its scores are unsigned: suppression mechanisms appear with the same sign as promotion. At the circuit level, path patching [Wang et al., 2023], ACDC [Conmy et al., 2023], and EAP-IG [Hanna et al., 2024] discover circuits by intervening on activations; some can identify composition pathways, but doing so requires separate interventions per edge, scaling linearly with circuit size. Circuit tracing [Ameisen et al., 2025] trains a cross-layer transcoder to build attribution graphs from sparse features, but requires per-model training. Our method produces signed contributions, decomposes attention through all three (K, Q, V) composition pathways, recovers composition structure as a natural byproduct of the backward recursion, and runs on the original model from a single forward pass without intervention or auxiliary training.

3 Method

3.1 Unified key-value framework

While multi-head self-attention and MLP appear to be distinct model components, they can both be viewed as key-value lookups.

For attention, this perspective is inherent in its design. Each position i computes a query $Q_i = X_i W_Q$ and compares it against keys $K_j = X_j W_K$ at all positions via $\langle X_i, X_j \rangle_{\text{QK}}$. The softmax over these scores produces a distribution that determines how much to retrieve from each position's value $U_j = X_j W_V$. The query selects which keys to match, and the corresponding values are written into the residual stream.

For MLP, Geva et al. [2021] proposed to view it in the same way by rewriting the MLP formula as

$$\text{MLP}(X) = \phi(X K^\top)\, U, \tag{4}$$

where the columns of W_{up} serve as keys and the rows of W_{down} as the corresponding values. The activation function ϕ acts as the selection mechanism, analogous to softmax in attention. We extend this analogy by noting that attention and MLP share the same $\phi(S)\, U$ structure (Table 1), and we use this unified view as the basis for a single backward attribution procedure that applies to both sublayers.

Table 1: Unified key-value format $\phi(S)\,U$ for attention and MLP.

	Attention	MLP
ϕ	softmax	ReLU / GELU
S	$QK^T/\sqrt{d_{\text{head}}}$	XW_{up}
U	XW_V	W_{down}

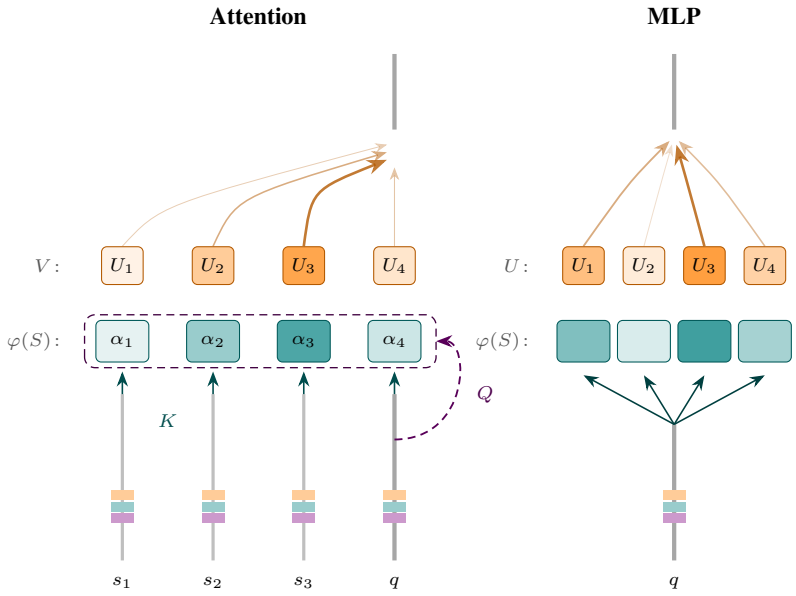


Figure 1: Unified key-value view. Both attention and MLP compute a weighted sum $\sum_i \varphi(S_i) \cdot U_i$: a selection mechanism $\varphi(S)$ (green, opacity indicates strength) weights values U (orange), and the result is added to the residual stream at position q. In attention, position q serves 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.

3.2 Key-side decomposition and scoring

Combining the unified KV framework with the residual-stream view, we can model interactions between residual components through key-side decomposition. Let c_{kj} denote the k-th residual component at position j, so $X_j = \sum_k c_{kj}$. By bilinearity of the attention logit,

$$\langle X_q, X_s \rangle_{\text{QK}} = \sum_k \langle X_q, c_{ks} \rangle_{\text{QK}}, \tag{5}$$

that is, with the query fixed, the attention logit at (q, s) decomposes additively across upstream components writing at s. Similarly, by linearity of the up-projection,

$$X_q W_{\text{up}} = \sum_k c_{kq} W_{\text{up}}. \tag{6}$$

Both attention and MLP thus decompose the corresponding S in Table 1 into per-component contributions.

We denote these contributions as $s_k^{\text{attn}}(q, s) := \langle X_q, c_{ks} \rangle_{\text{QK}}$ and $s_{k,j}^{\text{mlp}}(q) := c_{kq}\, W_{\text{up}}^j$, where W_{up}^j denotes the j-th column of W_{up}. Since we assume a pre-norm architecture, each sublayer's input passes through LayerNorm before projection. We handle this via *marginal normalization*: each component c_k is centered, divided by the full residual stream's standard deviation, and scaled by the LayerNorm weight before projection. We absorb this into the notation and write c_k for the normalized component throughout.

To quantify how strongly each upstream component drives the sublayer, we collapse s_k^{attn} and s_k^{mlp} to scalar scores. For attention, softmax is translation-invariant, so shifting all logits by a constant does not affect the resulting distribution. We use the standard deviation of the centered logit contributions:

$$\text{score}_k^{\text{attn}}(i) = \text{std}(s_k^{\text{attn}}(i)). \tag{7}$$

For MLP, the activations ReLU and GELU are magnitude-sensitive. Both positive entries (which activate keys) and negative entries (which suppress them) are meaningful, so we use the L2 norm:

$$\text{score}_k^{\text{mlp}}(i) = \|s_k^{\text{mlp}}(i)\|_2. \tag{8}$$

Knockout experiments validate that these scalar scores predict causal effect across the Pythia-deduped family from 160m to 6.9b parameters; details are deferred to Appendix E and Appendix F.

3.3 Recursive backward attribution

Section 3.2 showed how to decompose a single attention head's logit or a single MLP's keys into per-component contributions. Since each sublayer's output is a weighted sum of its values U_i, where each weight is determined by the corresponding score S_i that itself decomposes additively across upstream components, this gives a natural recursive procedure: at each component, distribute credit among the values by how much each contributes, then at each value distribute its share among the upstream components that drove its selection score; treating those as new parents and applying the same decomposition to each, we walk backward through the model until credit lands on the input embeddings. This section gives the mental model for the procedure. The full algorithm, with the soft floor that controls credit amplification, is in Appendix B.

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:

$$I(c_k) = \langle c_k(p),\, d_t \rangle, \qquad d_t := \frac{w_{\text{LN}}}{\sqrt{\text{Var}(X_p) + \epsilon}} \odot \big(W_U[t] - \overline{W}_U\big), \tag{9}$$

where $\overline{W}_U$ is the vocabulary-averaged unembedding vector. The centering ensures $I(c_k)$ measures how much c_k increases the target token's probability relative to the uniform baseline, rather than its contribution to the raw logit, since softmax is shift-invariant. We call $I(c_k)$ the component's *importance*, and the recursion's job is to redistribute each component's importance back to the upstream components that produced it, continuing until credit lands on input embeddings.

Inside a sublayer. Both attention and MLP write into the residual stream as $\phi(S)\,U = \sum_i \phi(S)_i \cdot U_i$. Forward, the value U_i at each key i is what moves on to do the downstream computation, weighted by how strongly the key is selected. When the credit comes back, we need to trace this same path in reverse: first distribute the parent's importance among the keys by how much each U_i contributes to the credit direction, then at each key distribute its share among the upstream components that drove $\phi(S)_i$ in proportion to their contributions to S_i. The first step picks which keys mattered, the second asks who made those keys selected. The attention and MLP rules below specialize this two-step pattern.

Attention dispatch. For an attention head at query position q, the keys are indexed by source position s, and each $U_s = X_s W_V W_O$ writes into the head's output weighted by $\alpha_{h,q,s}$. At the first step (depth 0), we split the parent's importance across source positions in proportion to $\alpha_{h,q,s} \cdot \langle U_s - \overline{U_s}, d_t \rangle$, the share each gated value contributes to the target direction d_t. A source whose value pushes the head's output toward d_t receives positive credit, and one that pushes against it receives negative credit. At deeper recursion steps, the credit direction is no longer d_t-aligned, so we fall back to the post-softmax attention weights $\alpha_{h,q,s}$ alone for the dispatch. At each source s, the credit further descends into upstream components writing the residual at s, weighted by their contributions to the attention logit $\langle X_q, X_s \rangle_{\text{QK}}$ on the source side. The recursion continues at position s with each upstream component as a new parent.

MLP dispatch. For an MLP, the keys are neurons. Each neuron's value $U_j = W_{\text{down}}^j$ is a constant parameter, weighted by the gate $\phi(\text{pre}_j)$. At the first step, we split the parent's importance across neurons by how much each neuron's gated value contributes to the target direction d_t, so inactive

neurons receive no credit. At deeper steps, where the credit direction is no longer d_t-aligned, we split by gate activation alone in the same spirit. At each neuron, the credit further descends into upstream components by their contributions to the neuron's pre-activation, which decomposes additively by linearity of the up-projection. Unlike attention, the MLP introduces no new position; the recursion continues at q with each upstream component as a new parent.

Termination and conservation. Source layers strictly decrease at every recursion step, so every path terminates at the embedding layer. Embeddings are terminal: their credit accumulates at the corresponding token position and contributes to the final per-token attribution. Aggregate credit is conserved up to the soft floor on the denominators (Appendix B.2), which prevents amplification when positive and negative contributions nearly cancel. We use a single value of this floor, $\beta = 0.8$, across all experiments.

Variants. The decomposition admits several principled choices on three independent axes, summarized in Table 2.

Axis 1: Attention key-side. Credit at each attention edge can descend through the K-branch alone or through all three K, Q, and V composition pathways. K-only treats the query as fixed context; K+Q+V traces credit through all three inputs that jointly determine the attention pattern.

Axis 2: MLP key-side. The deeper-recursion dispatch can follow the unified key-value view, weighting each upstream component by its gated contribution to the MLP output (Eq. 11), or step outside the key-value view and score each upstream contribution by its L_2 norm (Eq. 16).

Axis 3: V-side dispatch. Credit can be distributed by raw activation alone (post-softmax weights for attention; gate activations for MLP) or by output-aligned shares that weight each value by how much it projects onto the realized sublayer output. The aligned MLP rule requires the weighted MLP dispatch from Axis 2, since the L_2 variant does not decompose at the per-neuron level; aligned attention dispatch can be combined with either MLP rule.

The rules described in Sections 3.2 and 3.3 correspond to K-only + weighted + raw. The remaining five combinations are detailed in Appendix B.3 and compared empirically in Section 4.2.

4 Evaluation on IOI

4.1 Setup

The pipeline runs the backward attribution of Section 3.3 on each of 100 IOI prompts generated at seed 42. From a single forward pass per prompt, we extract token-level attribution (Section 4.2) and composition structure via rerooting (Section 4.3) on GPT-2 small, then extend the token-level analysis across the Pythia-deduped family from 160 M to 6.9 B parameters (Section 4.4).

Hyperparameters and configurations. We use $\beta = 0.8$ throughout; the trace pruning threshold is $\tau = 10^{-4}$. We test six configurations per model that span the three-axis variant design space of Section 3.3, summarized in Table 2. KQV variants use equal branch weights $w_K{=}w_Q{=}w_V{=}\frac{1}{3}$.

Table 2: Six configuration variants spanning the design space.

Configuration	Attn key-side	MLP key-side	V-side dispatch
`k_only_weighted`	K only	weighted	raw
`k_only_l2`	K only	L_2 norm	raw
`k_only_aligned`	K only	weighted	aligned
`kqv_weighted`	K+Q+V	weighted	raw
`kqv_l2`	K+Q+V	L_2 norm	raw
`kqv_aligned`	K+Q+V	weighted	aligned

4.2 Token attribution

Token-level attribution is obtained by summing all credit that reaches the embedding at each position; we report the result as a percentage of total positive credit across positions. For each of the

100 IOI prompts, we prepend the BOS token in front of each of the sentences to separate the effects of attention sink. We measure three metrics: *IO > S1*, the fraction of prompts where the IO token receives more credit than the first subject mention; *IO > S2*, the same against the second subject mention; and *IO top-1*, the fraction of prompts where IO is the single highest-attributed position.

Table 3: IOI token attribution across six configuration variants ($n = 100$, GPT-2 small, $w_K{=}w_Q{=}w_V{=}\frac{1}{3}$). **IO>S1**: fraction of prompts where IO receives more credit than S1. **Top-1**: IO is the highest-attributed position. **Top-1***: excluding BOS (position 0), which absorbs residual attention-sink credit in several configurations.

Configuration	IO>S1	IO>S2	Top-1	Top-1*	Mean IO
`k_only_weighted`	55%	57%	2%	38%	−0.1%
`k_only_l2`	70%	73%	1%	55%	+10.0%
`k_only_aligned`	78%	71%	21%	58%	+18.2%
`kqv_weighted`	99%	100%	64%	99%	+37.9%
`kqv_l2`	99%	100%	51%	98%	+30.7%
`kqv_aligned`	99%	100%	97%	98%	+39.9%

The results in Table 3 suggest that attention-side dispatching is the most significant factor for the algorithm. This is expected: the K-only variants trace credit through one of three input pathways, discarding Q and V composition entirely, so the gap reflects missing information rather than sensitivity to design choices. Specifically, all K+Q+V variants identify the IO token as the top-1 most important token much more frequently than the K-only configurations. Furthermore, the K+Q+V variants rank IO higher than both S1 and S2 in nearly all prompts ($\geq 99\%$), whereas K-only setups achieve this in fewer than 80% of the prompts.

The weighted and L2 variations are more sensitive to the attention sink effect, removing BOS tokens from comparison significantly increases the percentage that IO tokens are ranked first. In contrast, the aligned variants are robust against the attention sink. To understand why, we computed the cosine similarity between each source position's projected value and the realized attention output at the name mover heads across 10 randomly selected prompts. We found that the BOS token's projected value has near-zero alignment with the realized output (mean $+0.08$ across Name-Mover heads A9.H9, A9.H6, A10.H0), while the IO token achieves $+0.93$. Since the aligned decomposition weights each source by its projection onto the direction the head actually writes, the BOS contribution is naturally suppressed without explicit filtering.

Disentangling copying from circuit suppression. The IO>S metrics above are partially confounded by the copying mechanism: when the model predicts IO, credit naturally flows to the position where IO appears, regardless of whether the IOI circuit is active. To isolate the circuit's contribution, we design a three-way comparison at the *duplicate position* S2.

In IOI prompts, S2 is the second mention of the subject name, the position where the model detects the repeated name and triggers suppression of S in favor of IO. We decompose the logit difference $\ell_S - \ell_{\text{IO}}$ (i.e., target=S) and measure credit at S2. Two competing mechanisms act at this position: the copying mechanism promotes S (S2 *is* S), while the IOI circuit suppresses it. We compare against matched ABC prompts where S1 is replaced by a distinct name B and S2 by C, eliminating the duplicate signal:

- **C→C** (ABC, target=C): pure copying baseline, where C promotes itself with no opposing circuit.
- **S2→S** (IOI, target=S): copying and IOI suppression compete at S2.
- **C→B** (ABC, target=B): no copying signal, as C is unrelated to B.

Table 4: Credit at the S2/C position under three conditions ($n = 100$, GPT-2 small). **S1→S**: credit at S1 in the IOI condition for comparison. The IOI circuit reduces S2's self-promotion credit from the copying baseline (C→C) to near the level of a positionally matched but unrelated token (C→B), while S1, the first mention and not the duplicate trigger, retains strong copying credit.

Configuration	C→C	S2→S	C→B	S1→S	S1–S2 gap
`k_only_weighted`	+21.2	+8.8	+6.8	+11.9	+3.1
`k_only_l2`	+20.8	+8.1	+6.7	+12.4	+4.3
`k_only_aligned`	+26.3	+8.2	+9.4	+15.5	+7.3
`kqv_weighted`	+21.6	+6.6	+7.0	+20.4	+13.8
`kqv_l2`	+17.5	+5.7	+6.2	+17.8	+12.1
`kqv_aligned`	+23.9	+7.3	+8.7	+22.2	+14.9

Table 4 shows that in all configurations, the IOI circuit reduces credit at S2 from the copying baseline (C→C $\approx +20$) to near the positional baseline (C→B $\approx +7$). The self-promotion signal at S2 is almost entirely cancelled by the suppression mechanism. Meanwhile, S1, the first mention of the same name and not the duplicate trigger, retains strong copying credit. The S1–S2 gap measures this asymmetry: two occurrences of the same token in the same decomposition receive radically different credit because only S2 participates in the duplicate-detection circuit.

The K+Q+V configurations show markedly larger S1–S2 gaps (12–15) than the K-only variants (3–7), indicating that the V and Q channels carry suppression signal that the K-only decomposition does not capture. This is consistent with the token attribution results above. We adopt `kqv_aligned` for cross-scale experiments based on both analyses, and verify that results are robust to the SafeDenom parameter β (Appendix D).

4.3 Composition analysis

The recursion in Section 3.3 can start from any component, not just the target token. Given a component c at position p, **rerooting** traces backward from c's output, revealing which upstream components compose into c and through which branch (K, Q, or V). We use rerooting to test the three composition claims from Wang et al. [2023]:

1. **NM←S-Inh**: S-Inhibition heads modulate Name Mover queries [Wang et al., 2023, Sec. 3.2].
2. **S-Inh←Dup/Ind**: Duplicate Token and Induction heads write positional signal into S-Inhibition values [Wang et al., 2023, Sec. 3.3].
3. **Ind←Prev**: Previous Token heads feed Induction keys [Wang et al., 2023, Sec. 3.3].

Setup. For each claim, we reroot the tracer (`kqv_aligned`) at the head of interest at the appropriate position (END for claims 1–2, S2 for claim 3) and rank upstream attention heads (layer ≥ 1) by their credit in the composition mode stated by Wang et al. [2023]. We report two metrics:the fraction of rerootings where the expected upstream role ranks among the top-100 upstream components by aggregate path credit (*found%*), and the median rank of the first head from that upstream role when it does appear. Results are pooled over all heads of interest per claim (n=300 for 3 Name Movers, n=400 for 4 S-Inhibition or 4 Induction heads).

Table 5: Composition verification via rerooting (`kqv_aligned`, n=100, GPT-2 small). Each row uses the composition mode stated by Wang et al. [2023], filtered to attention heads layer≥1. Cells: median rank / found%. Bold = the claimed connection. Full matrices in Appendix A.

Mode	↓ / ↑	NM	S-Inh	Ind	Dup	Prev
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%**

All three claims are verified: in the expected composition mode, the predicted upstream heads appear in over 95% of rerootings, with median ranks at or below 3, indicating that they are among the strongest contributors through the stated pathway.

Beyond the three claims, the full ranking matrices (Appendix A) reveal additional composition structure. Duplicate Token heads dominate the Q-side input of Induction heads (median rank 1, found on 100% of rerootings), suggesting that the duplicate-detection signal reaches Induction heads by modifying what they query for. S-Inhibition heads receive strong K-side input from both Induction and Duplicate Token heads (median ranks 5 and 4, found on 93% and 91%), indicating that these upstream components shape which positions S-Inhibition attends to. Induction heads receive V-side input from Previous Token heads (median rank 9, found on 88%), consistent with their mechanistic role: Previous Token heads write positional information into the values that Induction heads read. A more detailed investigation of these additional composition patterns is left to future work.

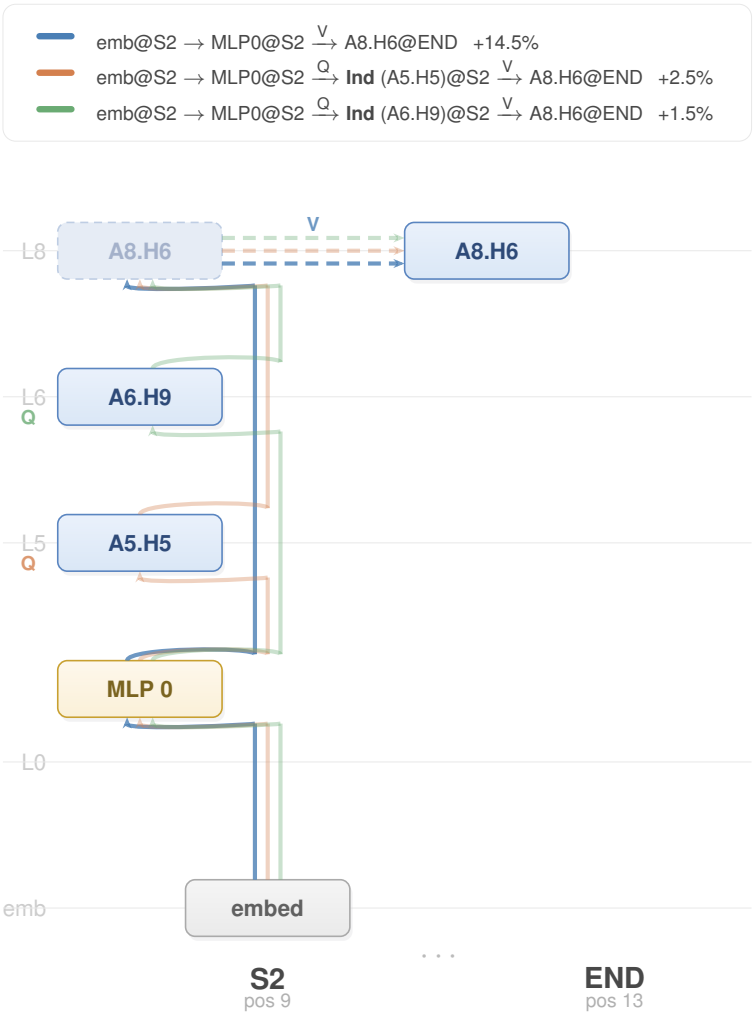


Figure 2: Top-3 composition paths from rerooting at S-Inhibition head A8.H6, filtered to V-mode (GPT-2 small, single prompt). The method outputs named paths such as `embed@S2` → `MLP0@S2` $\xrightarrow{V}$ `A8.H6@END` (blue, +14.5%), tracing the token embedding at position S2 through MLP 0 into the S-Inhibition head's value input at the END position via cross-position attention (dashed line). The second and third paths route through Induction heads A5.H5 and A6.H9 at position S2 before reaching A8.H6, confirming the Ind → S-Inh composition described by Wang et al. [2023]. Each hop is labeled with its composition mode (K, Q, or V).

4.4 Cross-scale analysis

We apply the method to five Pythia-deduped models (160M, 410M, 1.4B, 2.8B, 6.9B) using `kqv_aligned`, selected based on the configuration analysis on GPT-2 (Tables 3 and 4).

Token attribution. Table 6 reports IO ranking metrics across scales. From 410M onward, IO is the highest-credited token on every prompt across all three metrics. Even at 160M, where the model assigns only 19.5% probability to IO on average, the method ranks IO above both S1 and S2 on at least 83% of prompts.

Table 6: Token attribution across Pythia-deduped scales (`kqv_aligned`, $n = 100$, target=IO).

Model	IO>S1	IO>S2	Top-1	Mean IO	P(IO)
160M	83%	85%	76%	+35.2%	0.195
410M	100%	100%	100%	+69.4%	0.348
1.4B	100%	100%	100%	+62.9%	0.449
2.8B	100%	100%	100%	+60.4%	0.473
6.9B	100%	100%	100%	+48.9%	0.466

S2 suppression across scales. Table 7 reports the three-way S2 comparison. The S1–S2 gap is positive at every scale, confirming that the method distinguishes the duplicate-detection position (S2) from the first mention (S1) across the entire model family. S2 credit is near zero or negative at all scales except 2.8B, where the gap narrows and S2 retains moderate positive credit, suggesting a less structured IOI circuit at this scale.

Table 7: S2 suppression across Pythia-deduped scales (`kqv_aligned`, $n = 100$, target=S). S1–S2 gap measures the credit differential between two occurrences of the same name in the same decomposition.

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

5 Limitations

Absolute credit accumulation. The composition analysis ranks upstream components by total absolute path credit. At deep recursion, MLP layers and early attention heads accumulate large scores through path volume rather than mechanistic relevance. Mode-specific filtering (K, Q, or V) recovers the expected circuit structure, but this requires knowing which composition mode to examine. Developing ranking criteria that surface mechanistically relevant contributions without prior hypotheses is an open problem.

Evaluation scope. All experiments use the IOI task, a well-understood two-name completion task with known circuit structure. Whether the method generalizes to tasks with less structured or unknown circuits remains to be tested. In particular, IOI's relatively short prompts and clear functional roles make it an ideal but potentially unrepresentative testbed.

Single forward pass. The method decomposes the model's actual computation on a given input rather than measuring counterfactual effects. It reveals what the model *did* compute, not what it *would* compute under perturbation. This is complementary to interventional methods rather than a substitute.

Architecture assumptions. The marginal LayerNorm decomposition assumes a pre-norm architecture, as used by GPT-2 and the Pythia family. Post-norm models place LayerNorm after the

residual addition, which would require a different decomposition strategy. We do not test post-norm models in this paper.

Path explosion at depth. The number of paths grows combinatorially with recursion depth. The pruning threshold τ keeps enumeration tractable, but paths through many intermediate components carry increasingly diluted credit. The top-k paths stored per rerooting (2000 in our experiments) may miss relevant but low-magnitude composition routes, particularly in deeper models.

Computational scaling. The decomposition requires computing key-side contributions from every upstream component to every downstream sublayer. For a model with L layers, H heads per layer, and sequence length n, this involves $O(L^2Hn)$ score computations, each operating on d_{model}-dimensional vectors. Our implementation streams one source layer at a time, computing its contributions to all later layers before discarding its activations, which avoids storing the full $L \times L$ interaction matrix simultaneously. This keeps memory linear in the number of downstream layers rather than quadratic. Nevertheless, as d_{model} and L grow, the per-edge computation becomes the bottleneck: the 6.9B model already requires substantially longer preparation time than GPT-2 small. Extending to models beyond 10B parameters may require further optimizations such as low-rank approximations of the key-side projections or selective computation of only the highest-scoring edges.

6 Conclusion

We presented UNPACK, a non-interventional attribution method built on the observation that both attention and MLP follow a unified key-value template $\phi(S)\,U$. A single backward recursion through this structure produces interaction strengths between components, named end-to-end paths with K/Q/V composition labels, and per-token attribution from one forward pass.

On GPT-2 small, the method recovers all three composition connections in the IOI circuit described by Wang et al. [2023], and reveals that each connection routes through a specific composition mode. Token attribution distinguishes the duplicate-detection mechanism from trivial copying: two occurrences of the same name receive radically different credit because only the second triggers the IOI circuit. Across five Pythia-deduped models from 160M to 6.9B parameters, the suppression pattern holds at every scale, suggesting the method captures genuine circuit structure rather than artifacts of a particular model size. Applying the method to tasks beyond IOI, where ground-truth circuits are unavailable, is a natural next step.

References

Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 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/.

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.

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.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-NeoX-20B: An open-source autoregressive language model. In Angela Fan, Suzana Ilic, Thomas Wolf, and Matthias Gallé, editors, *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin, May

2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bigscience-1.9. URL https://aclanthology.org/2022.bigscience-1.9/.

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, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 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*, pages 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, 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.

Neel Nanda. Attribution patching: Activation patching at industrial scale, 2023. URL https://www.neelnanda.io/mechanistic-interpretability/attribution-patching.

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, page 3319–3328. JMLR.org, 2017.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In Yonatan Belinkov, Najoung Kim, Jaap Jumelet, Hosein Mohebbi, Aaron Mueller, and Hanjie Chen, editors, *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.blackboxnlp-1.25. URL https://aclanthology.org/2024.blackboxnlp-1.25/.

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, editors, *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 Composition ranking matrices

Tables 8 report the rank of the first upstream head from each role tier, filtered to attention heads above layer 0. Each cell shows median rank and the percentage of rerootings where the upstream role was found among the stored paths. Lower rank is stronger composition.

Table 8: Composition ranking matrices (`kqv_aligned`, n=100, GPT-2 small, attn heads layer$\geq$1). Each cell: median rank / found%. Lower rank = stronger composition. Rows = head of interest, columns = upstream role.

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%

B Algorithm details

In this section we give the details for the algorithm described in Section 3.3. We first introduce the SAFEDENOM soft floor used to stabilize signed normalization (Section B.2), then give the full backward recursion as pseudocode (Section B.1), and finally specify each variant of the three dispatch subroutines used in the recursion (Section B.3).

B.1 Pseudocode

Algorithm 1 gives the backward recursion. Three subroutines, ATTENTIONV, ATTENTIONK, and MLPDISPATCH, are referenced by name; each has a default implementation specified in Section B.3 along with its variants.

Algorithm 1 Recursive backward attribution

Require: target token t, position p, pruning threshold τ, amplification floor β (Eq. 10)
$d_t \leftarrow$ target direction (Eq. 9)
$I(c_k) \leftarrow \langle c_k(p),\, d_t \rangle$ for each residual component c_k at position p
RECURSE($\{(c_k, I(c_k), d_t)\}, p$)

Procedure RECURSE(scores, q)
for each (component c_k, importance I, credit direction d) in scores with $|I| \geq \tau$ **do**
  if c_k is the embedding **then**
    credit$[q]$ += I
  else if c_k is attention head (L, h) **then**
    $\{(s, I_s, d_s)\} \leftarrow$ ATTENTIONV(I, q, h, L, d)
    for each s with $|I_s| \geq \tau$ **do**
      $\{(c_j, I_j, d_j)\} \leftarrow$ ATTENTIONK(I_s, q, s, L, d_s)
      RECURSE($\{(c_j, I_j, d_j)\}_j, s$)
    end for
  else if c_k is MLP at layer L **then**
    $\{(c_j, I_j, d_j)\} \leftarrow$ MLPDISPATCH(I, q, L, d)
    RECURSE($\{(c_j, I_j, d_j)\}_j, q$)
  end if
end for

The pruning threshold τ keeps the recursion tree tractable by dropping sub-credits below τ in absolute value. This is necessary because the recursion expands by a factor of roughly $K \cdot S$ per level (number of components times sequence length); any nontrivial τ keeps path enumeration practical.

Aggregate token-level credit reported in Section 4 is computed via 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.2 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 by orders of magnitude. 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:

$$\textsc{SafeDenom}(\{r_i\}; \beta) := \text{sign}\Big(\sum_i r_i\Big) \cdot \max\Big(\big|\sum_i r_i\big|,\ \beta \cdot \sum_i |r_i|\Big). \tag{10}$$

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 D.

B.3 Variant details

The recursion's three subroutines correspond to the three independent variant axes of Section 3.3: ATTENTIONV maps to V-side dispatch (raw or aligned), ATTENTIONK to attention-side branches (K-only or K+Q+V), and MLPDISPATCH to the MLP key-side rule (weighted or L_2). The default configuration, K-only + weighted + raw, instantiates each subroutine with its first listed implementation below.

Value-side dispatch. Both ATTENTIONV and the weighted variant of MLPDISPATCH share a common rule: with parent credit I flowing in along direction d, distribute I across keys i in proportion to each value's projection on d, weighted by the gate:

$$I_i = I \cdot \frac{\phi(S)_i \cdot \langle U_i,\, d\rangle}{\textsc{SafeDenom}(\{\phi(S)_j \cdot \langle U_j,\, d\rangle\}_j, \beta)}. \tag{11}$$

For attention, $\phi(S)_s = \alpha_{h,q,s}$ and $U_s = X_s W_V W_O$ (with per-vector mean centered along the hidden dimension as in Section 3.2); keys are indexed by source position s, and the recursion continues at s with the credit direction unchanged ($d_s = d$). For MLP, $\phi(S)_j = \phi(\text{pre}_j)$ and $U_j = W_{\text{down}}[j, :]$ (centered analogously); keys are indexed by neuron j, and the recursion continues at q with credit direction $d_j = (\phi(\text{pre}_j)/\text{pre}_j) \cdot W_{\text{up}}[:, j]$, the effective direction in residual space along which neuron j absorbs information.

ATTENTIONV: V-side dispatch (raw vs. aligned). The default raw rule applies Eq. 11 as written. The output-aligned variant overrides the credit direction at the first call by replacing d_t with the realized attention output direction at the query,

$$\tilde{d} \;=\; o_{\text{attn}}(q) \;=\; \sum_h \sum_s \alpha_{h,q,s}\, U_s, \tag{12}$$

so that source-position credit at depth 0 is split by how much each gated value projects onto where attention actually wrote, rather than onto d_t. At deeper depths, d is inherited from the parent unchanged in both variants.

ATTENTIONK: attention-side branches. The default K-only rule decomposes the parent's source-side credit I_s into upstream components c_j writing the residual at position s, weighted by their key-side contributions to the attention logit $\langle X_q, X_s \rangle_{\text{QK}}$:

$$I_K(c_j, s) = I_s \cdot \frac{s_j^{\text{attn}}(q, s)}{\textsc{SafeDenom}(\{s_m^{\text{attn}}(q, s)\}_m, \beta)}, \tag{13}$$

where $s_j^{\text{attn}}(q, s) = \langle X_q, c_{js} \rangle_{\text{QK}}$. The recursion continues at s with d_j inherited from d_s. The K+Q+V variant adds two parallel branches at every attention edge, with branch weights w_K, w_Q, w_V summing to 1. The Q-branch decomposes the symmetric query-side bilinear form $s_j^{\text{attn,Q}}(q, s) = \langle c_{jq}, X_s \rangle_{\text{QK}}$ and recurses at the query position q:

$$I_Q(c_j, q) = I_s \cdot \frac{s_j^{\text{attn,Q}}(q, s)}{\textsc{SafeDenom}(\{s_m^{\text{attn,Q}}(q, s)\}_m, \beta)}. \tag{14}$$

The V-branch decomposes the head's value contribution at s, $s_j^{\text{attn,V}}(s) = c_{js} W_V W_O$, against the parent's credit direction d_s:

$$I_V^{(\text{branch})}(c_j, s) = I_s \cdot \frac{\langle s_j^{\text{attn,V}}(s),\, d_s \rangle}{\textsc{SafeDenom}(\{\langle s_m^{\text{attn,V}}(s),\, d_s \rangle\}_m, \beta)}. \tag{15}$$

We use default branch weights $(w_K, w_Q, w_V) = (0.333, 0.333, 0.333)$.

MLPDISPATCH: MLP key-side rule (weighted vs. L_2). The default weighted rule treats the MLP as a key-value lookup following Geva et al. [2021] and applies Eq. 11 with the MLP's keys and values as described above. After the value-dispatch step writes credit to each neuron j along its effective direction d_j, the recursion descends into upstream components c_k writing the residual at q, weighted by their key-side contributions $s_k^{\text{mlp}}(q)[j] = c_{kq} W_{\text{up}}[:, j]$ (each neuron treated independently; credit flowing through different neurons recombines at each upstream component).

The L_2 variant steps outside the key-value view of the MLP and instead scores each upstream component directly by the L_2 norm of its full key-side contribution, with sign inherited from the parent:

$$I_M^{L_2}(c_k) = I \cdot \text{sign}(I) \cdot \frac{\|s_k^{\text{mlp}}(q)\|_2}{\sum_m \|s_m^{\text{mlp}}(q)\|_2}. \tag{16}$$

The recursion continues at q. Because L_2 does not decompose the MLP at the per-neuron level, output alignment is undefined under this variant; we therefore test it only with the raw V-side rule.

C 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.

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.

Final LayerNorm. The pre-unembedding LayerNorm is folded into the target direction d_t in Eq. 9. We use the variance of the residual X_p at the time of the forward pass rather than running statistics, since d_t is computed per prompt and a running average would mix prompt-specific scales.

Logit-difference targets. For contrastive tasks such as IOI, we replace $W_U[t] - \overline{W}_U$ in Eq. 9 with $W_U[t] - W_U[t']$, where t' is the distractor token (e.g. the subject S in IOI). Vocabulary-mean centering is unnecessary in this case because the constant cancels in the difference. The resulting d_t is the gradient of the target-minus-distractor logit with respect to the residual at p, scaled by the LayerNorm factor.

Pre-norm assumption. The marginal-LayerNorm normalization in Section 3.2 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. We do not test post-norm models in this paper.

D Sensitivity to β

The SafeDenom parameter β controls the amplification floor when distributing credit through near-cancelling components (Section 3). We evaluate sensitivity on GPT-2 using 10 IOI prompts across $\beta \in [0, 2]$ with the `kqv_aligned` configuration, measuring both token attribution (target=IO) and the S2 suppression analysis (target=S).

Table 9: Attribution metrics across β values. Pairwise ranking metrics (IO>S1, IO>S2) are stable for all $\beta > 0$. Credit magnitudes increase with β as the amplification floor widens, but the structural pattern—S2 suppression relative to S1—is preserved throughout.

β	IO>S1	IO>S2	S1→S	S2→S	S1–S2 gap	Mean IO
0.0	70%	50%	−34.0	+3.3	−37.3	−0.6%
0.1	90%	100%	+8.8	+3.6	+5.2	+21.9%
0.2	100%	100%	+10.3	+3.9	+6.4	+22.1%
0.4	100%	100%	+12.8	+4.2	+8.6	+24.4%
0.6	100%	100%	+14.3	+4.5	+9.8	+28.3%
0.8	100%	100%	+15.1	+5.1	+10.0	+32.1%
1.0	100%	100%	+14.7	+5.1	+9.6	+34.3%
1.5	100%	100%	+15.2	+5.2	+10.0	+38.6%
2.0	100%	100%	+15.7	+5.4	+10.3	+42.2%

Pairwise ranking metrics (IO>S1, IO>S2) achieve 100% for all $\beta \geq 0.2$. The S1–S2 gap, which measures the credit differential between the first and second occurrence of the same name, stabilizes around $+10$ for $\beta \geq 0.6$. Credit magnitudes (Mean IO) increase monotonically with β because a larger floor dampens redistribution at cancellation points, concentrating credit at unambiguous positions. This systematic effect preserves relative rankings while shifting absolute scale. We use the library default $\beta = 0.8$ throughout.

E Knockout validation

Before applying the backward attribution of Section 3.3, we verify that the key-side scores from Section 3.2 are causally meaningful: do the scores predict what happens when a component is removed from the model? We test this across the Pythia-deduped family [Biderman et al., 2023] at five scales (160M, 410M, 1.4B, 2.8B, 6.9B), all at training step 143,000. Scores are computed over 1,000 sentences sampled from the Pile [Gao et al., 2020].

Setup. For each model, we run the scoring method on 1,000 Pile sentences. Every residual-stream component (the embedding, every attention head at every layer, and every MLP) receives two aggregated importance scores. The *attention strength* $\sum_{L,h,q} |\,\overline{s_k^{\text{attn}}}\,|$ sums absolute mean key-side scores over all downstream attention heads and queries. The *MLP strength* $\sum_{L,q} \|\,\overline{s_k^{\text{mlp}}}\,\|_2$ is the analogous quantity over downstream MLPs. To measure causal importance we perform a *communication-specific* ablation: on a clean forward pass we extract each component's residual vector c_k, then re-run the model with c_k subtracted from the input of one type of downstream LayerNorm (cut-attention or cut-MLP). We measure Δppl on 200 held-out Pile sentences (7,967 tokens). For tractability, for each source layer we select 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.

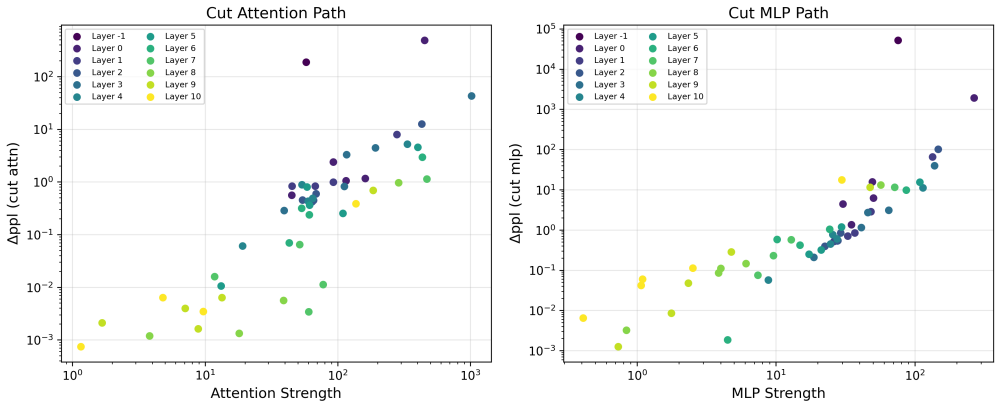
Results. Figure 3 plots importance score against causal knockout effect for each of the five models. The relationship is strongly monotonic on both channels and at every scale, spanning four to five orders of magnitude on each axis. Table 10 reports Spearman rank correlations: within-layer mean

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 color gradient from purple (early) to yellow (late), with early layers dominating the MLP channel and late-layer attention heads dominating the attention channel.

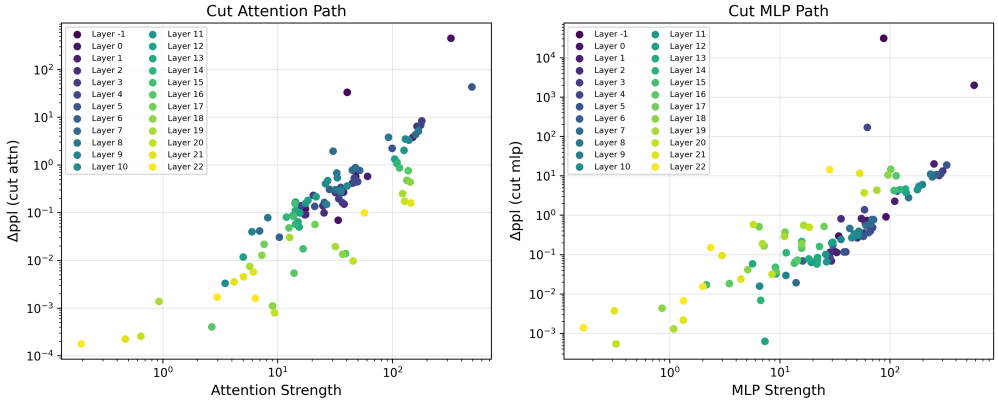
Table 10: Spearman rank correlation between importance score and knockout Δppl. *Within-layer* ρ is the mean of per-source-layer Spearman correlations (the stricter test, holding the set of downstream receivers fixed). *Cross-layer* ρ pools all components for a given channel.

		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

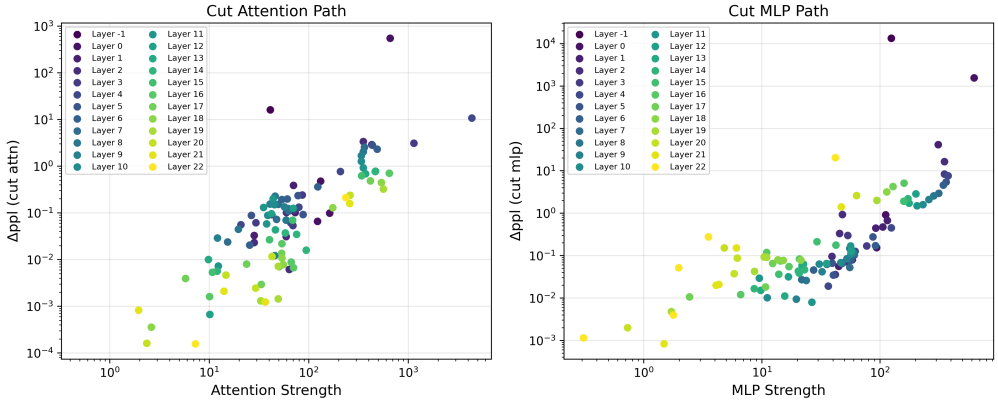
F Knockout figures



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

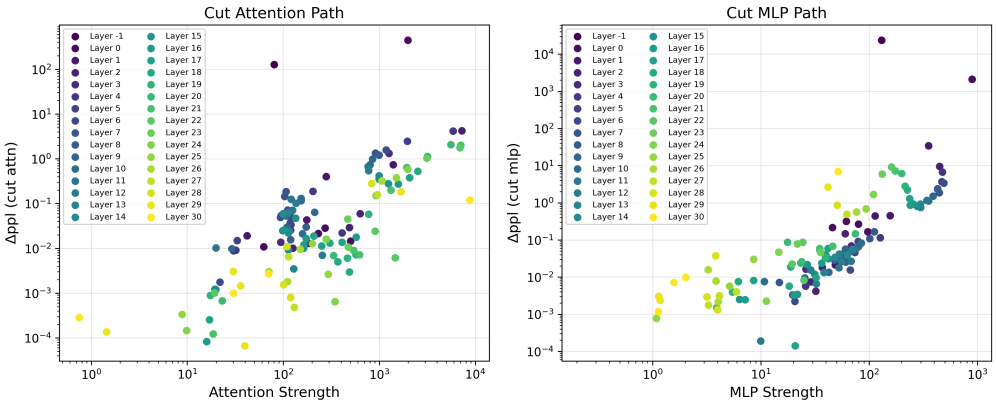


(b) Pythia-410M-deduped (24 layers, 116 knockouts).

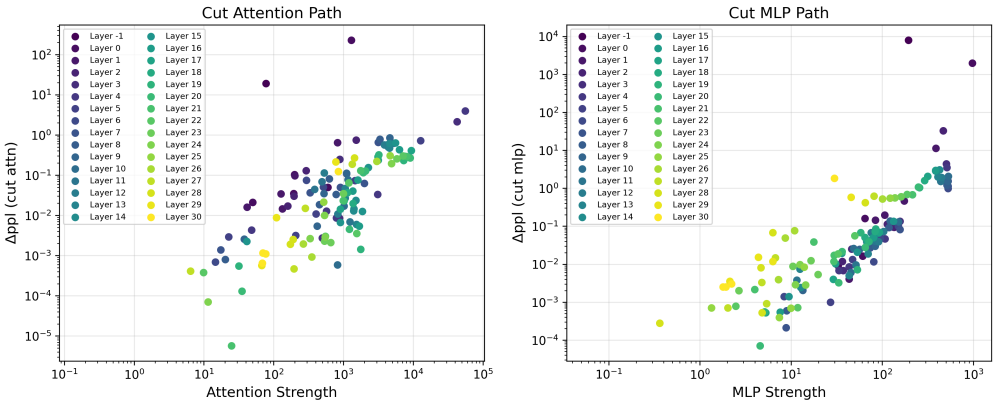


(c) Pythia-1.4B-deduped (24 layers, 116 knockouts).

Figure 3: Knockout validation across the Pythia-deduped family, part 1 of 2. Each point is one residual-stream component; the x-axis is our importance score, the y-axis is Δppl when the corresponding communication channel is ablated. **Left panels:** attention channel (score = attention strength, ablation = cut attention path). **Right panels:** MLP channel (score = MLP strength, ablation = cut MLP path). Points are colored by source layer (darker = earlier; layer -1 is the embedding). Both axes log-scaled. The monotonic score-to-Δppl relationship is stable across the $40\times$ parameter range covered in this figure and Figure 4.



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



(b) Pythia-6.9B-deduped (32 layers, 156 knockouts).

Figure 4: Knockout validation across the Pythia-deduped family, part 2 of 2. Axes, layer coloring, and channel layout as in Figure 3.