How Much Do Circuits Tell Us? Measuring the Consistency and Specificity of Language Model Circuits

Anonymous authors
Paper under double-blind review

Abstract

The circuits framework in mechanistic interpretability aims to identify sparse subgraphs of model components that are causally responsible for a behavior, typically evaluated by measuring *necessity* and *sufficiency*. But these criteria say little about whether a circuit consistently captures how a model performs a task, or if it is specific to that task. We study these two properties, *consistency* and *specificity*, across six tasks and five models, extracting circuits at the component level (attention heads and MLP blocks) and at the level of individual MLP neurons. We find that component-level circuits are highly consistent and causally important for most tasks, but they are not specific: for a given task, ablating its own circuit drops accuracy by about as much as ablating another task's circuit. Neuron-level circuits, on the other hand, exhibit higher specificity across tasks, but far less consistency within tasks. This is explained by circuit overlap: component-level circuits share most of their components across tasks, related or not, while neuron-level circuits overlap only between closely related tasks. In a case study of the components shared by the task circuits of `Llama-3.2-3B`, we show that they consist mostly of MLP blocks, while the few attention heads within turn out to have general-purpose roles. Overall, our findings raise questions about the degree to which circuits can support targeted understanding of, and intervention on, model behavior.

1 Introduction

Neural networks are infamously black box; even when we can elicit strong performance on a task, it is unclear which internal computations are responsible. Mechanistic interpretability seeks to reverse-engineer these computations by identifying *circuits*: sparse subgraphs of model components that are causally responsible for a particular behavior (Elhage et al., 2021; Wang et al., 2023). A growing body of work has developed methods for extracting such circuits (Syed et al., 2024; Marks et al., 2025; Jafari et al., 2025) and evaluating their *necessity* (removing the circuit should degrade performance) and *sufficiency* (the circuit alone should reproduce the behavior; Shi et al., 2024).

We argue that there are two additional properties crucial to consider (Figure 1). First, circuits should be *consistent*: most of the same units should recur across individual examples of a task, or else the aggregated task circuit is too noisy to describe the task. Second, circuits should be *specific*: a task's circuit should be meaningfully different from the circuits of unrelated tasks, since a circuit that surfaces for any task says little about how the model solves one task in particular.

We test both properties at scale, using two automated methods for circuit discovery: edge attribution patching with integrated gradients (EAP-IG; Hanna et al., 2024), and relevance patching (RelP; Jafari et al., 2025). For each method, we extract circuits at the component level (attention heads and MLP blocks) and at the level of individual MLP neurons (Arora et al., 2026), for n=1000 examples across six tasks spanning algorithmic reasoning (`Addition`, `Boolean Logic`), information retrieval (`IOI`, `CopyColors MCQA`), and knowledge-intensive benchmarks (`ARC Easy`, `ARC Challenge`). We study five models from three architecture families (`Gemma 2`, `Llama 3.2`, `Qwen3`), and additionally report results on `OLMo-2-1B` pre-training checkpoints. We find the following:

1. **Component-level circuits are consistent but not specific.** Across tasks and models, a large fraction of each per-example circuit is drawn from the same set of components, and ablating

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.

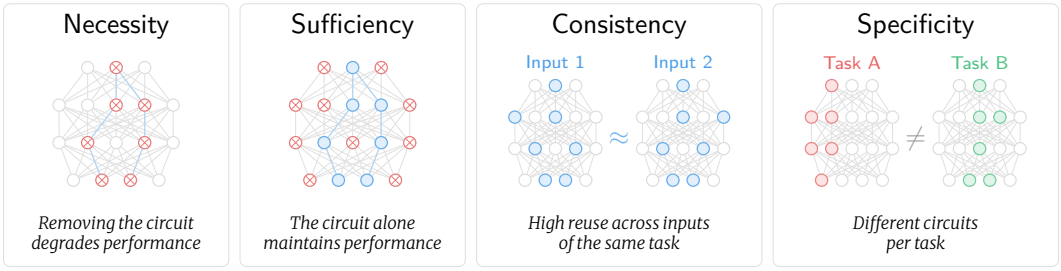


Figure 1: **Circuit evaluation criteria.** We propose evaluating circuits for consistency across inputs and specificity across tasks, in addition to necessity and sufficiency.

this set (knocking them out) causes larger accuracy drops than ablating a capacity-matched random set of the same size. But these components are not task-specific: ablating one task's circuit damages an unrelated task about as much as ablating that task's own circuit does. This is explained by the substantial overlap between component-level circuits, which are composed of largely the same MLP blocks.

2. **Neuron-level circuits are specific but not consistent.** Circuits derived from individual MLP neurons tell a different story. A task is damaged far more by ablating its own neuron-level circuit than by ablating any other task's circuit, but the neurons that recur across examples of a task cover only a small fraction of each per-example circuit. Because there are far more MLP neurons than transformer components, these circuits are also much smaller as a fraction of the model.

We conclude that component-level circuits are too coarse to isolate task-specific mechanisms, while neuron-level circuits are too fine-grained to characterize the task in general. We discuss several possible reasons for this - such as the role of shared MLP blocks and superposition/polysemanticity (Elhage et al., 2022) - and present a case study of the components shared across tasks in `Llama-3.2-3B`. We also find preliminary evidence that feature-level circuits (Dunefsky et al., 2024; Marks et al., 2025; Ameisen et al., 2025) do not recover these properties either (§F). Finally, we discuss implications for applications that assume circuit-level modularity, including model editing (Meng et al., 2022; Dai et al., 2022) and safety interventions (Li et al., 2023).

2 Background

2.1 Transformer Circuits

We adopt the residual-stream view of transformers introduced by Elhage et al. (2021), along with the computational graph formulation of Conmy et al. (2023), which represents decoder-only transformers as a directed acyclic computational graph. Under the Transformer Circuits view, the *residual stream* is interpreted as a shared communication channel running through the model. The token embeddings are written into it first, after which each attention head and MLP reads from the stream, performs a computation, and additively writes its output back. Because these contributions are additive, the output of any component can influence any downstream component, resulting in connections between non-adjacent layers. This naturally induces a graph in which nodes are connected whenever the output of one node can contribute to another. Following Elhage et al. (2021), nodes are commonly taken to be attention heads and MLP blocks. Recent work has introduced other decompositions, however, such as individual neurons or sparse autoencoder features, each with its own benefits (Marks et al., 2025; Ameisen et al., 2025; Arora et al., 2026). Edges represent the flow of information between components through the residual stream. A *circuit* is then a sparse subgraph of this computational graph that is responsible for a given behavior (Wang et al., 2023; Conmy et al., 2023).

2.2 Attribution Methods

To identify components which are important for a given behavior, researchers typically use activation patching (Vig et al., 2020; Meng et al., 2022; Wang et al., 2023), which compares a *clean* input

that elicits the behavior of interest with a corresponding *corrupted* input in which that behavior is different. Then, individual component's activation from the corrupted run are *patched* to the clean run. The resulting change is then measured and used to estimate the components' importance. However, this requires a separate forward pass per component, which is expensive. Edge attribution patching with integrated gradients (EAP-IG; Hanna et al., 2024) instead efficiently approximates the effect of patching a component by taking the product of its activation difference and the gradient of the metric, averaged over m points along the straight-line path between the corrupted and clean inputs. This allows all components to be scored simultaneously, making it suitable for the large-scale analysis we conduct here. Relevance patching (RelP; Jafari et al., 2025) computes the same product, but replaces the gradient with a layer-wise relevance propagation signal that conserves attribution from one layer to the next, requiring a single backward pass. See §B for full details of both methods.

3 Methodology

3.1 Tasks and Models

We evaluate six tasks spanning algorithmic reasoning (`Addition`, `Boolean Logic`), information retrieval from context (`IOI`; Wang et al., 2023, `CopyColors MCQA`; Mueller et al., 2025), and knowledge-intensive benchmarks (`ARC Easy`, `ARC Challenge`; Clark et al., 2018). We study five models from three architecture families: `Gemma 2` (2B, 2B IT; Gemma Team, 2024), `Llama 3.2` (3B, 3B Instruct; Llama Team, 2024), `Qwen3` (4B; Yang et al., 2025), as well as `OLMo-2-1B` (Walsh et al., 2025), which is used for pre-training analysis.

3.2 Circuit Discovery

For each task T, let $\mathcal{D}_T^{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n_{\text{train}}}$ denote the training set from which circuits are extracted, and let $\mathcal{D}_T^{\text{eval}}$ denote a disjoint held-out evaluation set on which all ablation accuracies are measured. Throughout, we use $n_{\text{train}} = 1000$ and $n_{\text{eval}} = |\mathcal{D}_T^{\text{eval}}| = 200$. Here, x_i denotes the input prompt and y_i the target answer token. These are paired with a corrupted input x_i' that minimally alters x_i, so that y_i is different.[1]

Let $\mathcal{C} = (\mathcal{V}, \mathcal{E})$ be the model's computation graph, where $\mathcal{V}$ is the set of model components and $\mathcal{E}$ are the connections between them. For notational simplicity, we identify a circuit with its selected node set; the corresponding subgraph is obtained by inheriting the relevant edges from $\mathcal{C}$. For each $(x_i, y_i) \in \mathcal{D}_T^{\text{train}}$, we extract a per-input circuit $\mathcal{C}_i \subseteq \mathcal{V}$ consisting of the top-$K\%$ of nodes according to their attribution scores, computed using EAP-IG or RelP. We sweep $K \in \{1, 5, 10, 20, 30\}$.

We instantiate $\mathcal{V}$ at two levels of granularity. At the *component* granularity, nodes are attention heads and whole MLP blocks. Our models have between 234 (`Gemma 2 2B`) and 1,188 (`Qwen3 4B` of these components). At the *neuron* granularity, nodes are the pre-down-projection MLP activations of each MLP block.[2] With d_{mlp} neurons per block, this gives between 229,376 and 350,208 nodes in total, corresponding to a node set approximately 300–1,000 times as large as at the component granularity.

Using the per-input circuits $\mathcal{C}_i$, we define a task's circuit at consensus threshold $P \in [0, 1]$ as

$$S_P = \left\{ c \in \mathcal{V} \,:\, \tfrac{1}{n_{\text{train}}} \textstyle\sum_{i=1}^{n_{\text{train}}} \mathbb{1}\{c \in \mathcal{C}_i\} \;\geq\; P \right\}. \tag{1}$$

Thus, S_P contains exactly the components that appear in at least $P \times 100\%$ of the per-input circuits.[3]

3.3 Within-Task Experiments

We define **Reuse@**P as the mean fraction of a per-input circuit overlapping with the task circuit,

$$\textbf{Reuse@}P = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \frac{|S_P \cap \mathcal{C}_i|}{|\mathcal{C}_i|}, \tag{2}$$

[1] Corruption strategies are described in §C.

[2] We discuss our motivation for this choice over other units, *e.g.*, transcoder latents, in §A, which is based on findings from Arora et al. (2026).

[3] We report P as a percentage in the text and figures.

Larger values indicate greater reuse of circuit components across examples at the specified consensus threshold. We sweep $P \in \{50\%, 75\%, 85\%, 90\%, 95\%, 96\%, 97\%, 98\%, 99\%, 100\%\}$, but report P=50% in the main text, since it is the only threshold at which the neuron-level circuit reuse is non-zero for most model-task pairs. The full sweep is reported in §K.

We next test whether the task circuit is causally important to task performance. We zero-ablate S_P and measure the accuracy drop on $\mathcal{D}_T^{\text{eval}}$, formalizing this via do-calculus (Pearl, 1995). For a circuit $S \subseteq \mathcal{V}$, let $\text{do}(S \leftarrow 0)$ denote the intervention that clamps the activations associated with every $c \in S$ to zero.[4] The model's output distribution on input x under this intervention is $p_{\mathcal{M}}(\cdot \mid x;\, \text{do}(S \leftarrow 0))$. Taking $S = \varnothing$ recovers the output distribution of the unablated model.

We define the **zero-ablated prediction** (ZAP) of $\mathcal{M}$ under ablation set S to be the model's maximum-probability token under the intervention:[5]

$$\text{ZAP}(\mathcal{M}, S, x) = \arg\max_{y'} p_{\mathcal{M}}\big(y' \mid x;\, \text{do}(S \leftarrow 0)\big), \tag{3}$$

Accuracy of the model on task T under ablation S is then [6]

$$\text{acc}_T(\mathcal{M}, S) = \frac{1}{n_{\text{eval}}} \sum_{(x,y)\, \in\, \mathcal{D}_T^{\text{eval}}} \mathbb{1}\{y = \text{ZAP}(\mathcal{M}, S, x)\}. \tag{4}$$

However, removing any set of components will reduce network capacity, so some degradation is expected even when the ablated components are not task-relevant. To account for this, we compare the consensus-circuit ablation against a **capacity-conserved control**, denoted S_{C^3}. The control is randomly sampled from $\mathcal{V} \setminus S_P$ and matched to S_P both in cardinality and in component-type composition. The difference in accuracy drop between S_P and S_{C^3} should therefore capture the effect of ablating the task circuit beyond degradation lost due to capacity. We define the causal effect of S_P on task T as

$$\text{necessity}(\mathcal{M}, S_P) = \frac{\text{acc}(\mathcal{M}, S_{C^3}) - \text{acc}(\mathcal{M}, S_P)}{\text{acc}(\mathcal{M}, \varnothing)}. \tag{5}$$

A positive necessity means ablating S_P hurts more than ablating C^3, which we interpret as evidence that the components shared across examples are *causally important* for task T.

3.4 Cross-Task Experiments

The experiments above establish whether a task's circuit is consistent across inputs and causally important for that task, but they do not tell us whether that circuit is *specific* to that task. To probe specificity, we conduct three experiments. First, we compare the effect of ablating a task's own circuit with the effects of ablating other tasks' circuits. Second, we quantify the overlap between pairs of task circuits. Finally, we partition pairs of circuits into shared and task-exclusive components to localize any task-specific causal signal.

First, define $\Delta_A^B = \text{acc}_A(\mathcal{M}, \varnothing) - \text{acc}_A(\mathcal{M}, S_P^B)$, the accuracy drop on task A when ablating task B's circuit, where acc_A denotes accuracy evaluated on $\mathcal{D}_A^{\text{eval}}$ and S_P^B is task B's circuit. If circuits are task-specific, ablating task A's own circuit should hurt task A more than ablating task B's circuit. For each task and model, we therefore compare Δ_A^A against $\frac{1}{|\mathcal{T}|-1} \sum_{B \in \mathcal{T} \setminus \{A\}} \Delta_A^B$, the mean accuracy drop on A when ablating each other task's circuit. We summarize this comparison with the *specificity gap*, $\Delta_A^A - \frac{1}{|\mathcal{T}|-1} \sum_{B \in \mathcal{T} \setminus \{A\}} \Delta_A^B$, measured in percentage points of accuracy. A gap near zero indicates that ablating a task's own circuit is no more damaging to that task than ablating other tasks' circuits.

We then measure the similarity between two tasks' circuits using the Jaccard index $J(S_P^A, S_P^B) = |S_P^A \cap S_P^B| \,/\, |S_P^A \cup S_P^B|$, which is 1 when the two circuits are identical and 0 when they are disjoint.

[4] Zero-ablation is one of several methods used in the literature (Wang et al., 2023; Zhang & Nanda, 2024; Li & Janson, 2024). In §E we repeat our experiments using mean ablation and find that our conclusions still hold.

[5] We use argmax decoding here, but any decoding rule could in principle be applied to $p_{\mathcal{M}}$.

[6] S, acc, and necessity all depend implicitly on T through $\mathcal{D}_T^{\text{train}}$ and $\mathcal{D}_T^{\text{eval}}$. We drop T from the notation for readability.

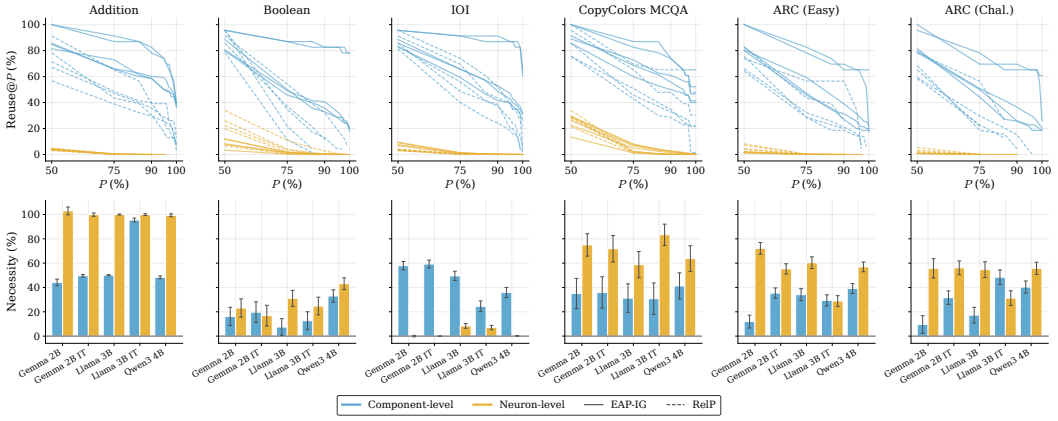


Figure 2: **Within-task circuit reuse and necessity.** *Top:* **Reuse**@P at top-K=10%, the fraction of each example's circuit covered by components that appear in at least P% of examples, as P sweeps from 50% to 100%. Curves show different models and attribution methods. *Bottom:* Necessity of circuits (at top-K=10% and P=50%) measures the difference in accuracy drop between ablating the task circuit and ablating the control set (S_{C^3}), with 95% C.I.'s.

Finally, we ask whether any task-specific causal signal is concentrated in components unique to a task or in components shared across tasks. For each task pair (A, B), we partition $S_P^A \cup S_P^B$ into three disjoint sets: the *shared core* $S_P^A \cap S_P^B$, the A*-only* set $S_P^A \setminus S_P^B$, and the B*-only* set $S_P^B \setminus S_P^A$. We ablate each set separately and measure its effect on task A. We then report the accuracy drop on the target task alongside the corresponding mean drop on the other tasks. We also compare the A-only set with a capacity-conserved control, constructed analogously to S_{C^3} above. Unlike our other experiments, we run these at P=100% rather than P=50%, and only at the component granularity, since at the neuron granularity the shared core is nearly empty, whereas component-level circuits remain non-empty even at P=100%.

4 Within-Task Consistency

We now present the results of our within-task experiments, looking first at circuit reuse and necessity, before examining the structure of these circuits, and how they evolve during `OLMo-2-1B` pretraining.

Component-level circuits reuse the same components across inputs. If a circuit captures how a model solves a task, most of the same components should appear regardless of which input is used. Figure 2 (top row) shows **Reuse**@P, the average fraction of a per-example circuit that is covered by the task's consensus circuit (Eq. 2). At K=10% and P=50%, component-level circuits show high reuse across inputs, with a median **Reuse**@P of 88.6% under EAP-IG and 78.4% under RelP. Reuse falls as P becomes more stringent: under EAP-IG the mean is 60.8% at P=85% and 38.5% at P=100% (where a component must appear in every example's circuit). Across the broader top-K and P sweeps, reuse remains high: under EAP-IG the median rises to 98.0% at K=30%, and no model–task pair falls below 50.0% (Figures 22 and 23).

Neuron-level circuits (mostly) do not reuse the same neurons across inputs. At the level of individual MLP neurons, per-example circuits are far less consistent. At K=10% and P=50%, neuron-level circuits show little reuse across inputs, with a median **Reuse**@P of 3.9% under EAP-IG and 4.5% under RelP (Figure 2). Neuron-level reuse is below 2% by P=75% and effectively zero by P=85% (Figure 2), and remains much lower than component-level reuse at every circuit size for both attribution methods (§K and Figure 22). Although neuron-level reuse is low throughout, it does vary, with `CopyColors MCQA` and `Boolean Logic` showing higher reuse than others (Figure 2).

Circuits are often causally important, but this varies by task. Figure 2 (bottom row) shows the necessity of each task's circuit (Eq. 5). At K=10% and P=50%, necessity is clearly positive for `IOI`, `Addition`, `CopyColors MCQA`, `ARC Easy`, and `ARC Challenge`, but is low for `Boolean Logic`

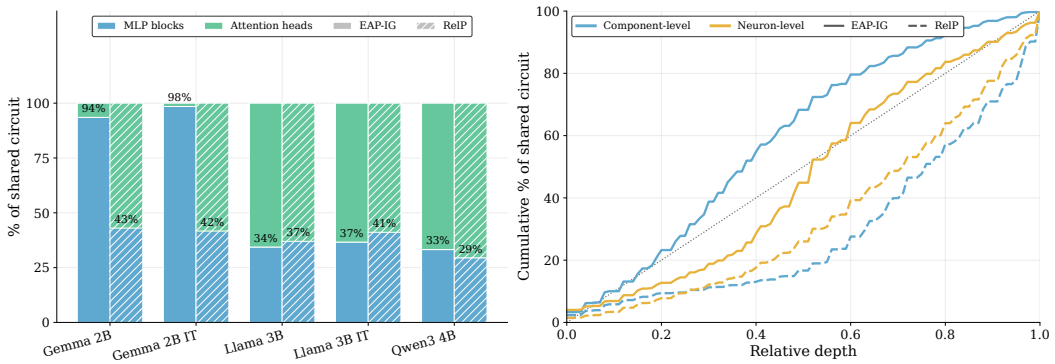


Figure 3: **Composition and depth of task circuits at K=10%, P=50%.** *Left:* the fraction of each component-level circuit made up of MLP blocks and attention heads. *Right:* the cumulative fraction of circuits at or below each relative depth. See §G.6 for per-model and per-task versions.

at both granularities and for `IOI` at the neuron level.[7] Therefore, reuse alone does not necessarily establish that a consensus circuit matters for the task. Neuron-level circuits exhibit far lower reuse than component-level circuits, but are still causally important at $P = 50\%$, with higher necessity than component-level circuits.[8] Conversely, component-level circuits have high reuse but low necessity for `Boolean Logic`. The full sweeps across top-K and P are reported in §K.

Different attribution methods find different circuits. EAP-IG and RelP scores are only weakly correlated (mean Spearman $\rho = 0.36$ between the the two methods' component scores on the same example), and their circuits for the same task overlap with a Jaccard index of only 0.23 at the component level and 0.16 at the neuron level (K=10%, P=50%, per model and task in Figure 13), less than the EAP-IG circuits of two different tasks (0.56, §L). The two methods' circuits also differ in structure (Figure 3). Relative to their base rates, MLPs are substantially overrepresented (*e.g.*, , in `Llama-3.2-3B`, they are only ∼4% of nodes but up to 41% of a circuit), most of all in the EAP-IG circuits of the `Gemma-2` models (93.5% and 98.5% MLP blocks). Circuits are mostly distributed in the later layers, although EAP-IG draws more from earlier layers than RelP (see §G for sizes, composition tables and per-model depth distributions).

Consistency emerges early and degrades over training. Tracking circuit reuse across `OLMo-2-1B`'s full pretraining run shows that reuse is highest within the first 32B tokens and declines over the rest of pretraining (§J and Figure 21). We also measure necessity, but find it hard to interpret as baseline accuracy is around chance for most of training. See §J for full results.

5 Cross-Task Specificity

We have shown that component-level circuits are consistent while neuron-level circuits are not, but that circuits at both granularities are causally important. However, a component may be critical for task A only because it is critical for every task, revealing little about how the model performs A itself. We now ask whether the circuits identified for one task are *specific* to that task.

Neuron-level circuits are task-specific, while component-level circuits are not. If a circuit is specific to its task, ablating it should damage that task more than any other task's circuit. Figure 4 (bottom) plots, for each task and model, the drop from ablating that task's own circuit against the mean drop from ablating the other five tasks' circuits, across granularity and attribution method. We observe that component-level circuits under EAP-IG cluster along the diagonal (specificity gap of 0.7 points), and just above it for RelP (specificity gap of 5.4 points). In contrast, neuron-level circuits fall well above the diagonal, with specificity gaps of 29.4 points for EAP-IG and 26.8 points for RelP. The exception is `IOI`, whose neuron-level circuit leaves `IOI` accuracy unchanged when ablated.

[7] All models have above chance baseline accuracy (Table 3)

[8] At $P = 100\%$, neuron-level circuits are nearly empty (Figure 14), causing necessity to drop from about 51 points at $P = 50\%$ to 5–19 at $P = 85\%$ (Figure 28).

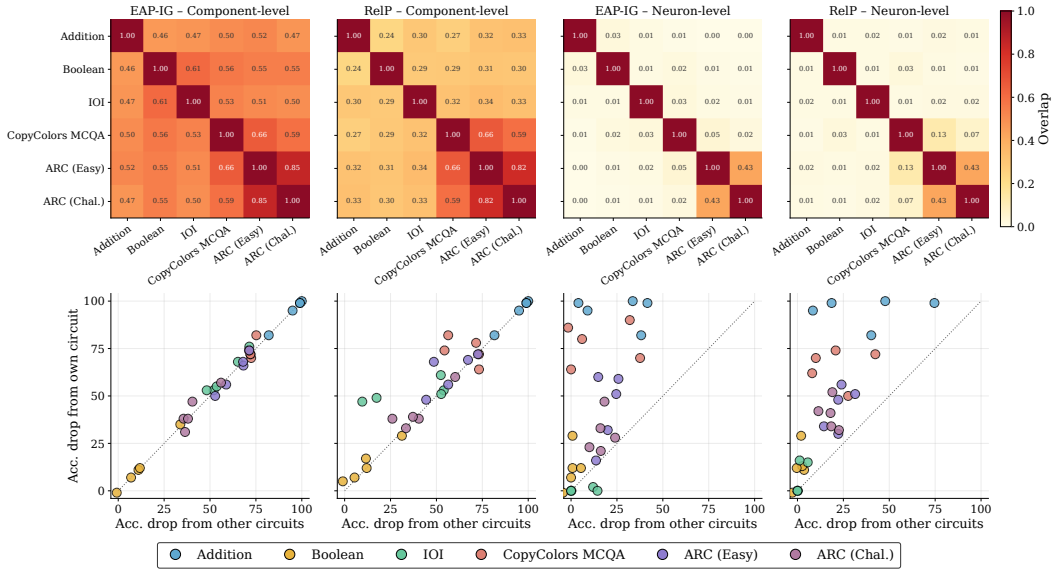


Figure 4: **Cross-task overlap and accuracy drops at K=10%, P=50%.** *Top:* Jaccard overlap between each pair of tasks' circuits, averaged over the five models. *Bottom:* For every model and task, we plot the accuracy drop from ablating that task's own circuit against the mean accuracy drop from ablating the other tasks' circuits. Per-model overlap across K and P and drops at every K are in §L.

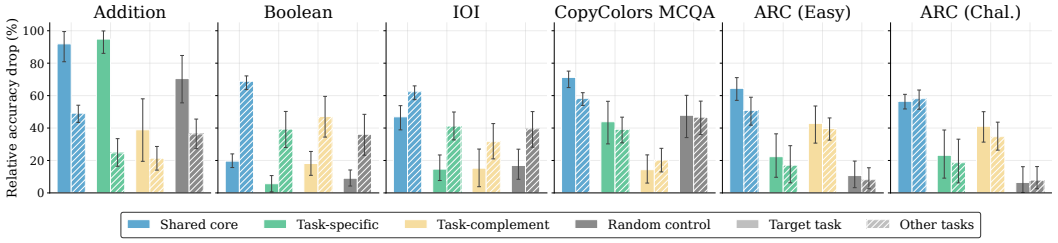


Figure 5: **Selective ablation of the component-level circuit partitions at K=10%, P=100%.** Solid bars represent the drop in accuracy on the target task (relative to baseline performance) and hatched bars the mean drop on the other five, averaged across models. See §H for all values of K.

Specificity is correlated with overlap between different task's circuits. Suppose two tasks' circuits were to overlap heavily. Then, if we ablated either circuit, we would expect this to cause similar drops in performance on the same task. We find that this is indeed the case: Figure 4 (top) shows the Jaccard overlap between each pair of tasks' circuits at the component- and neuron-level. Component circuits overlap heavily under EAP-IG, and under RelP they have smaller but still substantial overlap. For neuron circuits however, overlap is near zero for nearly every pair, with the exception of `ARC Easy` and `ARC Challenge`, which is unsurprising given that they come from the same benchmark.

Components exclusive to a task's circuit are not strongly task-specific. Although there is high overlap between component-level tasks, some components remain exclusive to a single task. To test whether these components pass our specificity check, we split each pair of task circuits into a shared core, a task-specific set, and a task-complement set, and ablate each one on its own (Figure 5). Ablating the shared core causes the largest drop on every task except `Addition`, consistent with our prior findings of low specificity. The task-specific set, meanwhile, damages the other tasks about as much as its own or more, and for `Boolean Logic`, `IOI` and `CopyColors MCQA` it even has low necessity. `Addition` is the only exception, where the task-specific set removes 95% of accuracy on its own task against 25% on the others. Conceptually, we hypothesize that these components are either noise from the attribution method, or that they are relevant to the task but not sufficient on their own to support it (see §H for more details).

Sparse feature circuits in `Gemma-2-2B` are similar to neuron circuits. We extend our analysis beyond components and neurons to sparse autoencoder latents (*i.e.*, features), repeating our experiments on `Gemma-2-2B` with latents as the circuit units, following Marks et al. (2025) and using the Gemma Scope SAEs (Lieberum et al., 2024). At K=10% and P=50%, feature circuits are only slightly more consistent than neuron circuits and both are far less consistent than component circuits. Feature circuits also match neuron circuits in necessity on every task and are as task-specific, with an average specificity gap of 28.3% of baseline accuracy against 29.7% for neurons, while component circuits show no gap. We discuss full results in §F.

5.1 Case study of shared attention heads in `Llama-3.2-3B`

We now conduct a basic case study on `Llama-3.2-3B`. Intersecting the task circuits under EAP-IG at K=10% and P=50%, we recover a set of 17 components, of which 14 are MLP blocks and three are attention heads. One of the heads, L4H10, is an attention sink (Xiao et al., 2024), sending 7.9 times more attention to the beginning-of-text token than to the body of the prompt, while the other two attend to task-relevant tokens regardless of the task. This is preliminary evidence that the shared core contains general-purpose machinery, though we leave an analysis of the MLP blocks and other models to future work. We include full details in §I.

6 Discussion

6.1 Why do circuits overlap?

MLP blocks as shared infrastructure. In our `Llama-3.2-3B` case study, 14 of the 17 components shared by all six task circuits are MLP blocks. These layers plausibly perform general-purpose operations – storing parametric knowledge (Sun et al., 2025; Liu et al., 2025), mapping tokens into a useful representational space, adjusting positional information – that all downstream computation depends on, regardless of task. Indeed, activation steering typically operates on post-MLP residual stream states (Subramani et al., 2022; Adila et al., 2026) rather than individual attention heads. More recently, for activation-based confidence estimation probes, features are also extracted from post-MLP states (Subramani et al., 2026).

Superposition and granularity. Elhage et al. (2022) observe that neural networks represent more features than they have dimensions. Thus, two tasks that rely on different features in the same attention head or MLP block will inevitably share that component. Neuron-level circuits (a finer-grained unit of computation) are substantially more task-specific, though only a small fraction of individual neurons recur across examples of the same task. One might expect feature-level circuits derived from SAEs or CLTs (Dunefsky et al., 2024; Marks et al., 2025; Ameisen et al., 2025) to recover both properties. However, we find preliminary evidence against this for `Gemma-2-2B`, where SAE feature circuits are as task-specific as neuron circuits and only slightly more consistent (§F).

Reuse as a feature, not a bug. So far, we have treated non-specificity as a limitation of circuit discovery, but this assumes that task circuits ought to be distinct. It could instead be the case that models develop (via training pressures) reusable computational motifs that support many different behaviors. Such reuse could plausibly contribute to generalization; *e.g.*, in-context learning likely works because models can apply the same retrieval and binding operations across novel contexts (Olsson et al., 2022; Arditi, 2026). Under this view, the shared core is a meaningful subject of study in and of itself.

6.2 What does this mean for circuit-level analysis?

Neither granularity (component or neuron) recovers both desired properties. Our results show that component- and neuron-level circuits seem to capture different properties of model computation. Component-level circuits are stable across inputs but overlap substantially across tasks, while neuron-level circuits are task-specific but vary across examples. A component-level circuit may therefore say little about what distinguishes one task from another, while a neuron-level circuit may not characterize the task as a whole. Our preliminary results with sparse feature circuits for `Gemma-2-2B` suggest that the same holds for feature-level circuits (§F). Claims about a circuit for a behavior therefore depend heavily on (among other things) the granularity at which it is studied.

Circuit structure depends on the attribution method. The main contrast between component- and neuron-level circuits holds under EAP-IG and RelP. However, the methods recover different amounts of component reuse and produce circuits with different compositions and layer distributions. This suggests that circuits obtained by different attribution methods will have qualitatively different structures.

Targeted interventions require finer notions of localization. The broad reuse of causally important components may have implications for model editing. Methods that modify particular weight matrices to target factual associations (Meng et al., 2022; Dai et al., 2022) may affect other behaviors if the same matrices support many tasks. This is consistent with evidence that causal localization does not straightforwardly inform editing (Hase et al., 2023) and that editing techniques can have low specificity (Hoelscher-Obermaier et al., 2023). Our results apply less directly to safety interventions that modify directions within activation space rather than entire components (Li et al., 2023; Zou et al., 2023; Lee et al., 2025).

7 Related Work

Circuit discovery and evaluation. One of the first works to almost fully reverse-engineer a model behavior was Wang et al. (2023), who identified an attention-only circuit for indirect object identification (IOI) in `GPT-2 Small`. Subsequent work has focused on scaling circuit discovery (Conmy et al., 2023; Syed et al., 2024). We use EAP-IG (Hanna et al., 2024) and RelP (Jafari et al., 2025) throughout for their scalability. On the evaluation side, Shi et al. (2024) propose statistical tests for *necessity*, *sufficiency*, and *minimality*, while Miller et al. (2024) show that standard evaluation metrics can be fragile. Our work adds *consistency* and *specificity* to this evaluation toolkit.

Circuit reuse across tasks. Several works have studied the concept of circuit reuse across tasks. Merullo et al. (2024) show that the IOI circuit in `GPT-2 Medium` is largely reused for a Colored Objects task, with 78% of its attention heads in common, Lan et al. (2024) find that related sequence continuation tasks rely on a shared sub-circuit, and Mondorf et al. (2025) find that functionally similar string-edit operations have overlapping circuits that can be composed to represent more complex operations. These works interpret shared components as evidence that the tasks share an algorithm. However, they do not disambiguate whether a shared component reflects a shared algorithm or is simply needed for every task (*i.e.*, specificity). Our cross-task experiments are designed to disambiguate between these cases, at two levels of granularity. We find that component-level circuits overlap even between unrelated tasks (*e.g.*, `Addition` and `ARC`) and are not specific to their task, whereas neuron-level circuits are, which suggests that shared components need not reflect an intuitive shared algorithm.

8 Conclusion

We demonstrate that consistency and specificity are important properties to consider for circuit discovery. Although circuits at the component and neuron level pass necessity tests, neither satisfies both properties. Component-level circuits are consistent but not specific, since circuits for different tasks overlap heavily and damage other tasks as much as their own. Neuron-level circuits are the opposite, with few neurons recurring across a task but far more damage to their own task than to any other, and sparse feature circuits in `Gemma-2-2B` behave similarly. This complicates the assumption that circuits are stable and unique descriptions of a task, and implies that controlling model behavior via circuits is not a precise intervention.

What should the field take from this? We think circuit discovery can still be useful, but researchers should be much more careful about what claims these techniques can support (since passing necessity tests alone does not imply other desired properties, such as consistency or specificity). Our findings also prompt several exciting questions for future work. Can we design attribution methods that explicitly incorporate some notion of specificity? Can high reuse on one task predict whether a model generalizes to other related tasks? Should we rethink the definition of a circuit, for example as a "soft" or "fuzzy" subgraph that varies across inputs rather than a single fixed one? And finally, what other properties of circuits should we care about, if we want to measure how useful circuits are as an abstraction for understanding model behavior?

AI USE STATEMENT

We used AI tools, specifically Claude Code, for writing the Matplotlib code used to make the plots in this paper, with human guidance on the specific style and content of said plots. We have verified that this code uses the data we collected from our experiments and creates accurate plots. Additionally, we used large language models, specifically Claude Fable 5.1, to review our final manuscript (i.e. by prompting it with "You are reviewer 2, please provide a thorough review of the attached paper"), which helped us to improve readability of the paper, but did not specifically lead to any new experiments being run. We otherwise have not relied on AI tools for any of the experiments or writing in this paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Dyah Adila, John Cooper, Alexander Yun, Avi Trost, and Frederic Sala. Weight updates as activation shifts: A principled framework for steering. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=eMCp3enKXM.

Ameen Ali, Thomas Schnake, Oliver Eberle, Grégoire Montavon, Klaus-Robert Müller, and Lior Wolf. XAI for transformers: Better explanations through conservative propagation. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 435–451. PMLR, 2022.

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Andy Arditi. In-context learning of representations can be explained by induction circuits. In *ICLR Blogposts 2026*, 2026. URL https://iclr-blogposts.github.io/2026/blog/2026/iclr-induction/.

Aryaman Arora, Zhengxuan Wu, Jacob Steinhardt, and Sarah Schwettmann. Language model circuits are sparse in the neuron basis. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=OrviwFWcN4.

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.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 82318–82355. Curran Associates, Inc., 2025. doi: 10.52202/085713-2755. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/764ff7477b8e24dbe01531f6791e8bdf-Paper-Conference.pdf.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018. URL https://api.semanticscholar.org/CorpusID:3922816.

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.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=J6zHcScAo0.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/to_model/index.html.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Gemma Team. Gemma 2: Improving open language models at a practical size. *ArXiv*, abs/2408.00118, 2024. URL https://api.semanticscholar.org/CorpusID:270843326.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=TZ0CCGDcuT.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=EldbUlZtbd.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching, 2024. URL https://arxiv.org/abs/2404.15255.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. Detecting edit failures in large language models: An improved specificity benchmark. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 11548–11559, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.733. URL https://aclanthology.org/2023.findings-acl.733/.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Farnoush Rezaei Jafari, Oliver Eberle, Ashkan Khakzar, and Neel Nanda. RelP: Faithful and efficient circuit discovery via relevance patching. In *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=5PKPy82sWN.

Michael Lan, Philip Torr, and Fazl Barez. Towards interpretable sequence continuation: Analyzing shared circuits in large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 12576–12601, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.699. URL https://aclanthology.org/2024.emnlp-main.699/.

Bruce W. Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. Programming refusal with conditional activation steering. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Oi47wc10sm.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, 2023.

Maximilian Li and Lucas Janson. Optimal ablation for interpretability. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=opt72TYzwZ.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, Janos Kramar, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In Yonatan Belinkov, Najoung Kim, Jaap Jumelet, Hosein Mohebbi, Aaron Mueller, and Hanjie Chen (eds.), *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 278–300, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.blackboxnlp-1.19. URL https://aclanthology.org/2024.blackboxnlp-1.19/.

Jiarui Liu, Jivitesh Jain, Mona Diab, and Nishant Subramani. Llm microscope: What model internals reveal about answer correctness and context utilization. *arXiv preprint arXiv:2510.04013*, 2025.

Llama Team. The llama 3 herd of models. *ArXiv*, 2024. URL https://api.semanticscholar.org/CorpusID:271571434.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I4e82CIDxv.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Circuit component reuse across tasks in transformer language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fpoAYV6Wsk.

Joseph Miller, Bilal Chughtai, and William Saunders. Transformer circuit evaluation metrics are not robust. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=zSf8PJyQb2.

Philipp Mondorf, Sondre Wold, and Barbara Plank. Circuit compositions: Exploring modular structures in transformer-based language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14934–14955, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.727. URL https://aclanthology.org/2025.acl-long.727/.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fried Fiotto-Kaufman, Tal Haklay, Michael Hanna, Jing Huang, Rohan Gupta, Yaniv Nikankin, Hadas Orgad, Nikhil Prakash, Anja Reusch, Aruna Sankaranarayanan, Shun Shao, Alessandro Stolfo, Martin Tutek, Amir Zur, David Bau, and Yonatan Belinkov. MIB: A mechanistic interpretability benchmark. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=sSrOwve6vb.

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

Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995. ISSN 00063444, 14643510. URL http://www.jstor.org/stable/2337329.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders, 2024. URL https://arxiv.org/abs/2407.14435.

Claudia Shi, Nicolas Beltran-Velez, Achille Nazaret, Carolina Zheng, Adrià Garriga-Alonso, Andrew Jesson, Maggie Makar, and David Blei. Hypothesis testing the circuit hypothesis in LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=5ai2YFAXV7.

Xiangchen Song, Aashiq Muhamed, Yujia Zheng, Lingjing Kong, Zeyu Tang, Mona T. Diab, Virginia Smith, and Kun Zhang. Mechanistic interpretability should prioritize feature consistency in sparse autoencoders. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2172–2210, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.99. URL https://aclanthology.org/2026.acl-long.99/.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. Extracting latent steering vectors from pretrained language models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 566–581, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.48. URL https://aclanthology.org/2022.findings-acl.48/.

Nishant Subramani, Palash Goyal, Yiwen Song, Mani Malek, Yuan Xue, Tomas Pfister, and Hamid Palangi. The ACUTE protocol: Operationalizing language model activations for better calibration, utility, and trust. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=67lpPXbYf6.

ZhongXiang Sun, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Weijie Yu, Yang Song, and Han Li. ReDeEP: Detecting hallucination in retrieval-augmented generation via mechanistic interpretability. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ztzZDzgfrh.

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.

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 (eds.), *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 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/.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12388–12401. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92650b2e92217715fe312e6fa7b90d82-Paper.pdf.

Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James Validad Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious (COLM's version). In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=2ezugTT9kU.

Kevin Ro 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 *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Hf17y6u9BC.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2023.

APPENDIX CONTENTS

A Limitations **16**

B Attribution Method Details **16**

B.1 Edge Attribution Patching 16

B.2 EAP with Integrated Gradients 16

B.3 Relevance Patching 17

C Task Details **17**

D Model Details **18**

E Robustness to the Ablation Method **18**

E.1 Mean ablation 19

E.2 Experimental Setup 19

E.3 Component-level circuits 20

E.4 Neuron-level circuits 20

F Sparse Feature Circuits **20**

F.1 Setup 21

F.2 Consistency and Necessity 21

F.3 Specificity 22

F.4 Circuit size 23

F.5 Dictionary size and attribution method 23

G Circuit Size and Composition **25**

G.1 Agreement Between Attribution Methods 25

G.2 Circuit Size 25

G.3 Component Types 25

G.4 Component Counts by Model 25

G.5 Cross-Task Circuit Decomposition Sizes 25

G.6 Layer Distribution 33

H Selective Ablation Across K **33**

I Case Study of Shared Components in `Llama-3.2-3B` **37**

J Pretraining Dynamics **38**

K Full Within-Task Results **39**

L Full Cross-Task Results **46**

A LIMITATIONS

Granularity. We analyze circuits at two levels of granularity throughout the main text: attention heads and MLP blocks, and individual MLP neurons. We also run our experiments over sparse autoencoder features for `Gemma-2-2B`, which has a public SAE release, and find the same tradeoff (§F). We do not run additional SAE experiments with the other four models, nor with cross-layer transcoders (CLTs; Ameisen et al., 2025).

Our primary reason is the computational cost associated with training such SAEs or CLTs (one per layer for each model we study, 108 in total for the remaining four). Furthermore, these feature bases are not necessarily "natural" to the model in the way the neuron basis is, and feature quality is not guaranteed. Indeed, prior work has identified issues such as feature splitting/absorption and (lack of) feature consistency, which may confound findings obtained from feature circuits (Chanin et al., 2025; Song et al., 2026).

Finally, recent work suggests that, for the purpose of circuit discovery, MLP neuron circuits are just as sparse as feature circuits, implying that they provide no added benefit (Arora et al., 2026). Our result using `Gemma-2-2B` is consistent with theirs: feature circuits are more consistent than neuron circuits on five of six tasks but far less so than component circuits, and they are as specific. Whether this holds across models remains an open question.

Task complexity. Our tasks, while diverse, do not include generative or multi-step tasks. More complex tasks may show different trends for consistency and specificity.

Model scale. We only study models up to 4B parameters. The tradeoff we observe may persist or disappear at larger scale, where a model may have enough capacity to represent each task with a circuit that is both stable across its examples and distinct from other tasks.

B ATTRIBUTION METHOD DETAILS

B.1 EDGE ATTRIBUTION PATCHING

Given a clean input x and a corrupted input x', EAP computes the attribution score for each component c as a first-order approximation of the effect of patching that component's activation:

$$\hat{e}_c = (a_c(x') - a_c(x))^\top \cdot \frac{\partial L(x)}{\partial a_c}$$

where $a_c(\cdot)$ denotes the activation of component c and $L(x)$ is a scalar metric evaluated on the clean input, here the log-probability of the correct answer (§C), and the score is summed over sequence positions. In practice, this is just a single line of PyTorch code:

```
score = (act_corrupt - act_clean) * grad
```

where `act_corrupt` and `act_clean` are the component's activations under the corrupted and clean inputs respectively, and `grad` is the gradient of the metric with respect to the clean activation. Per-component scores are obtained by summing over positions and (for attention heads) the head dimension.

B.2 EAP WITH INTEGRATED GRADIENTS

The first-order approximation above is unreliable wherever $\partial L/\partial a_c$ is close to zero at the clean activation, since then the score is small regardless of how much the activation itself matters (Hanna et al., 2024). EAP-IG replaces the single gradient with an integrated gradient (Sundararajan et al., 2017) accumulated along the straight-line path between the two inputs (x and x'). Defining z and z' as the token embeddings of x and x', the EAP-IG score of component u is

$$\hat{e}_c^{\mathrm{IG}} = (a_c(x') - a_c(x))^\top \cdot \frac{1}{m} \sum_{k=0}^{m-1} \frac{\partial L\big(z' + \frac{k}{m}(z - z')\big)}{\partial a_c},$$

where m is the number of integration steps. We use $m = 5$, the value Hanna et al. (2024) recommend.

B.3 Relevance Patching

RelP modifies attribution patching to better approximate the effects of actually patching activations (Jafari et al., 2025). It scores components with the same product of activation difference and gradient as attribution patching, but computes the gradient through a *replacement model*, whose forward pass is the same but has a modified backwards pass, in the manner of layer-wise relevance propagation, (Bach et al., 2015; Ali et al., 2022; Jafari et al., 2025). The score of component c is

$$\hat{e}_c^{\mathrm{RelP}} = (a_c(x) - a_c(x'))^\top \cdot \frac{\partial L_{\mathrm{rep}}(x')}{\partial a_c},$$

where L_{rep} denotes the metric computed through the replacement model. Gradients are computed using the replacement rules in Table 1. We evaluate the gradient on the corrupted input, corresponding to patching clean activations into the corrupted run. Thus, in the notation of Jafari et al. (2025), x' corresponds to the original input and x to the patch input.

We follow the replacement rules described by Arora et al. (2026), using the patching direction specified above. The normalization scale and attention pattern are treated as constants during backpropagation. For a differentiable elementwise activation σ with $\sigma(0) = 0$, we use

$$\widetilde{\sigma}(u) = u \odot \mathrm{Freeze}(r_\sigma(u)), \qquad r_\sigma(t) = \begin{cases} \sigma(t)/t, & t \neq 0, \\ \sigma'(0), & t = 0, \end{cases}$$

where r_σ is applied elementwise and Freeze treats its argument as a constant during backpropagation. For SiLU, this simplifies to $u \odot \mathrm{Freeze}(\mathrm{sigmoid}(u))$. For gated MLPs, we halve the gradients through the product using the LRP half rule. All other operations are unchanged.

Operation	Original expression	Replacement expression
RMSNorm	$\gamma_i x_i / \sqrt{\epsilon + \overline{x^2}}$	$\gamma_i x_i / \mathrm{Freeze}\left(\sqrt{\epsilon + \overline{x^2}}\right)$
Attention	$\sum_k A_{qk} v_k$	$\sum_k \mathrm{Freeze}(A_{qk})\, v_k$
Activation function	$\sigma(u)$	$\widetilde{\sigma}(u)$
Gated MLP product	$\sigma(u) \odot v$	$\frac{1}{2}\widetilde{\sigma}(u) \odot v + \frac{1}{2}\mathrm{Freeze}(\widetilde{\sigma}(u) \odot v)$

Table 1: **Replacement rules for RelP.** Each replacement preserves the original forward value while modifying the gradient computation. Freeze denotes the stop-gradient operator, and γ_i is the learned RMSNorm gain. In the gated-product row, u and v are the gate and up-projections of the MLP input, and $\widetilde{\sigma}$ includes the activation replacement defined above.

C Task Details

Activation patching and its approximations require a clean input x and a corrupted input x' that alters the information the model must use while preserving surface structure as much as possible. The attribution metric we use is the log-probability the model assigns to the correct answer, summed over the answer's token positions. Below we describe each task and its corruption strategy.

Addition. The model is given a 2-digit arithmetic problem (*e.g.*, "Compute: 47 + 63 =") and must output the correct sum. Corrupted inputs are generated by pairing each problem with a randomly selected different problem from the same batch, so that the corrupted input has the same format but different operands and answer.

Boolean Logic. The model is tasked with evaluating logical expressions composed of `and`, `or`, and `not` over boolean values (*e.g.*, "Evaluate: true and (false or true) =" → "true"). To construct a corrupted input, we randomly select a literal whose modification ((*e.g.*, `or` → `and`) would flip the expression's truth value. This ensures the corrupted input has the opposite answer while minimally changing the input.

Indirect Object Identification (IOI). Following the task setup in Wang et al. (2023), the model must identify the indirect object in sentences with a specific template involving two names (*e.g.*, "Bilbo and Frodo spoke in Rivendell before Bilbo gave Sting to" → "Frodo"). We use the dataset and counterfactuals from the Mechanistic Interpretability Benchmark (Mueller et al., 2025). The corrupted input applies the S2-IO flip counterfactual, which swaps the subject and indirect object names so that the correct completion changes while the sentence template remains the same.

CopyColors MCQA. This is a multiple-choice task from Mueller et al. (2025) where the model is provided a passage describing objects and their colors, followed by a question asking which color corresponds to a particular object. The answer choices are given as labeled options (A, B, C, D). The corrupted input is the answer-position counterfactual from the benchmark, which permutes the order of the answer choices so that the correct answer appears at a different position, changing the correct label token while keeping the passage and question unchanged.

ARC Easy / ARC Challenge. The AI2 Reasoning Challenge (Clark et al., 2018) consists of multiple-choice science exam questions. The Easy split contains questions answerable with basic retrieval and reasoning, while the Challenge split filters for questions requiring more complex inference. We use the datasets and counterfactuals from Mueller et al. (2025). As with CopyColors MCQA, the corrupted input applies the answer-position counterfactual, permuting the order of answer choices so that the correct label changes.

D Model Details

See Table 2 for a list of model information and Table 3 for each model's unablated accuracy on each task.

Model	**HuggingFace ID**	**Params**	$\|L\|$	$\|a\|$	d_{model}
`Gemma 2 2B`	`google/gemma-2-2b`	2.6B	26	8	2304
`Gemma 2 2B Instruct`	`google/gemma-2-2b-it`	2.6B	26	8	2304
`Llama-3.2-3B`	`meta-llama/Llama-3.2-3B`	3.2B	28	24	3072
`Llama-3.2-3B Instruct`	`meta-llama/Llama-3.2-3B-Instruct`	3.2B	28	24	3072
`Qwen3-4B`	`Qwen/Qwen3-4B`	4.0B	36	32	2560
`OLMo-2-1B`	`allenai/OLMo-2-0425-1B`	1.2B	16	16	2048

Table 2: **Models studied in this work.** We report the number of parameters, number of layers $|L|$, total number of attention heads $|a|$, and hidden dimension d_{model} for each model.

Model	Addition	ARC (Chal.)	ARC (Easy)	Boolean	IOI	CopyColors MCQA
`Gemma 2 2B`	80	57	82	64	100	100
`Gemma 2 2B Instruct`	96	77	92	57	100	98
`Llama-3.2-3B`	100	69	85	72	100	100
`Llama-3.2-3B Instruct`	99	76	88	66	100	98
`Qwen3-4B`	98	84	94	88	100	100

Table 3: Baseline accuracy (%) of each model on each task, measured on $\mathcal{D}_T^{\text{train}}$.

E Robustness to the Ablation Method

Our experiments measure necessity and specificity using zero ablation, which sets the output activation of each ablated component to zero. However, zero is an arbitrary value to the model (i.e. there's no reason to expect that zero is a neutral/baseline state for component activations), so various works have explored other ablation methods (Zhang & Nanda, 2024; Li & Janson, 2024).

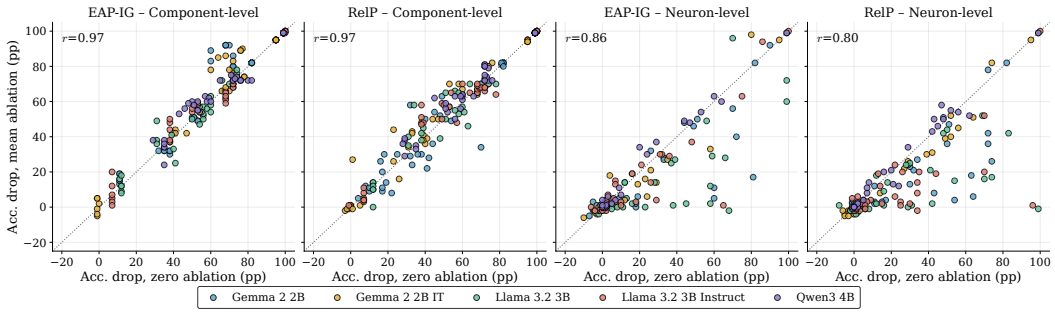


Figure 6: **Accuracy drop under zero and mean ablation.** One point per task pair (the drop on one task when the other's circuit, or its own, is ablated) at top-$K = 10\%$ and $P = 50\%$.

A popular alternative is mean ablation, which replaces the activation with its mean over some reference distribution of activations, such as the counterfactual/corrupted inputs (Wang et al., 2023). Other methods include resample ablation (replacing the activation with its value on a corrupted/counterfactual input; Heimersheim & Nanda (2024)), optimal ablation (replacing the activation with a constant chosen to minimize the model's loss; Li & Janson (2024)), and Gaussian noising (corrupting the input embeddings with Gaussian noise; Meng et al. (2022)). To understand whether our findings are robust to the choice of ablation method, we re-run the main experiments of §4 and §5 under mean ablation, using the average activation over the corrupted inputs of the target task. Alternatively, one could ablate on a per-input basis, replacing each component's activation on a given input with its activation on that input's paired counterfactual/corrupted input (i.e., resample ablation).

E.1 Mean ablation

Let $S \subseteq \mathcal{V}$ be the circuit to ablate, and let $\mathcal{D}_T^{\text{corr}}$ be the corrupted inputs of the target task. For each $c \in S$ and each token position i, let $\bar{a}_{c,i}$ denote the mean activation of c at position i over the inputs in $\mathcal{D}_T^{\text{corr}}$ of length at least i, and let $\bar{a}_S := (\bar{a}_{c,i})_{c \in S, i}$ denote the resulting family of mean activations.

Mean ablation is defined as the intervention $\text{do}(S \leftarrow \bar{a}_S)$, which clamps the activation of every $c \in S$ at each position i to $\bar{a}_{c,i}$. Positions beyond the longest corrupted input receive the mean over all positions, following Wang et al. (2023). The mean-ablated prediction and accuracy are then Eqs. (3) and (4) with $\text{do}(S \leftarrow 0)$ replaced by $\text{do}(S \leftarrow \bar{a}_S)$.

E.2 Experimental Setup

At the component level we use the EAP-IG and RelP circuits obtained at top-$K = 10\%$ and P of 50, 75, 85 and 100%. At the neuron level we use a P of 50, 75 and 85%, since neuron-level circuits are empty at $P = 100\%$, for the five main models. In Figure 6 we show the accuracy drops of every task pair at top-$K = 10\%$ and $P = 50\%$, the values used in §4 and §5, and in Table 4 we report correlations between mean and zero ablation over the full sweep of P.

		P=50%			P=75%			P=85%			P=100%		
Granularity	Method	Drop	ν	σ	Drop	ν	σ	Drop	ν	σ	Drop	ν	σ
Components	EAP-IG	0.97	0.96	-0.35	0.97	0.97	0.40	0.97	0.97	0.58	0.90	0.96	0.89
Components	RelP	0.97	0.95	0.86	0.96	0.91	0.96	0.96	0.93	0.96	0.91	0.93	0.94
Neurons	EAP-IG	0.86	0.96	0.93	0.98	0.99	1.00	0.97	0.99	0.99	–	–	–
Neurons	RelP	0.80	0.94	0.85	0.97	0.98	0.98	0.91	0.97	0.97	–	–	–

Table 4: **Agreement between zero and mean ablation.** Pearson correlations between accuracy drops under mean and zero ablation across task pairs ($n = 180$ per condition), and between the corresponding necessity (ν) and specificity gap (σ) values across model–task pairs ($n = 30$). All results use top-$K = 10\%$.

E.3 Component-level circuits

The findings of §4 and §5 hold under mean ablation for component-level circuits. Accuracy drops under mean and zero ablation are highly correlated for both EAP-IG and RelP ($r = 0.97$ for both at $P = 50\%$, Figure 6, and between 0.90 and 0.97 at the other values of P), and so is necessity (r between 0.91 and 0.97 at every P, Table 4). Necessity is slightly higher under mean ablation on average (0.59 against 0.56 for EAP-IG, 0.14 against 0.11 for RelP, Table 5).

For the specificity gap, EAP-IG circuits are near zero under both ablations (0.7 and 1.6 points on average, Table 5), so at low P the gaps carry little signal to correlate and the corresponding entries of Table 4 are uninformative (but included for completeness); at $P = 100\%$ they correlate ($r = 0.89$). RelP circuits have a small positive gap under both ablations (5.4 and 6.9 points), and their gaps are correlated across the two ablations at every P (r between 0.86 and 0.96).

	EAP-IG comp.		RelP comp.		EAP-IG neurons		RelP neurons	
Model	zero	mean	zero	mean	zero	mean	zero	mean
Necessity								
Gemma 2B	0.47	0.53	0.11	0.12	0.57	0.56	0.52	0.49
Gemma 2B IT	0.63	0.68	0.08	0.10	0.41	0.42	0.50	0.47
Llama 3B	0.49	0.53	0.07	0.09	0.51	0.46	0.48	0.39
Llama 3B IT	0.51	0.54	0.21	0.23	0.42	0.46	0.49	0.36
Qwen3 4B	0.69	0.68	0.08	0.15	0.53	0.55	0.51	0.51
Mean	0.56	0.59	0.11	0.14	0.49	0.49	0.50	0.45
Specificity gap (accuracy points)								
Gemma 2B	0.4	1.4	13.1	16.2	26.0	34.2	17.9	29.3
Gemma 2B IT	1.1	1.7	13.7	12.8	31.4	35.6	34.0	33.9
Llama 3B	-0.6	2.9	0.1	2.0	19.9	32.2	17.5	26.4
Llama 3B IT	0.2	1.7	1.1	-0.3	26.4	34.4	27.9	27.7
Qwen3 4B	2.7	0.4	-1.0	3.5	43.5	44.1	36.7	36.3
Mean	0.7	1.6	5.4	6.9	29.4	36.1	26.8	30.7

Table 5: **Necessity and specificity gap under zero and mean ablation** at top-K=10% and P=50%, averaged over the six tasks. Necessity is Eq. 5 and the specificity gap is in accuracy points.

E.4 Neuron-level circuits

At the neuron level, the findings of §4 and §5 also hold under mean ablation. Accuracy drops are correlated across the two ablations ($r = 0.86$ for EAP-IG and $r = 0.80$ for RelP at $P = 50\%$, and 0.91 to 0.98 at $P = 75$ and 85%), and necessity and the specificity gap are strongly correlated for both methods at every P (r between 0.85 and 1.00, Table 4). Neuron circuits are about as necessary under mean ablation as under zero ablation (0.49 against 0.49 for EAP-IG, 0.45 against 0.50 for RelP), and somewhat more specific (gaps of 36.1 against 29.4 points for EAP-IG and 30.7 against 26.8 for RelP, Table 5). The largest differences are in the RelP circuits of the two `Llama-3.2-3B` models, whose necessity is lower under mean ablation (0.36 and 0.39 against 0.49 and 0.48), and on `Addition`, whose specificity gap is larger under mean ablation in `Gemma-2-2B` and both `Llama-3.2-3B` models.

Overall, our findings about necessity and specificity **hold** under mean ablation at both granularities and for both attribution methods.

F Sparse Feature Circuits

We repeat the experiments of §4 and §5 using sparse autoencoder (SAE) latents as the circuit units, following Marks et al. (2025). An SAE is trained to reconstruct a model activation, in our case the residual stream after a layer, using an overcomplete set of learned latents (Bricken et al., 2023; Huben et al., 2024). The latent activations are trained to be sparse[9], typically through an L_1 penalty (Bricken

[9]See Elhage et al. (2022) and Bricken et al. (2023) for the motivation behind sparsity.

et al., 2023; Huben et al., 2024), although other methods exist (*e.g.*, TopK (Gao et al., 2025) and JumpReLU (Rajamanoharan et al., 2024)). Latents are commonly interpreted as features used by the model during its computation, and thus we refer to circuits using these latents as *feature circuits* (Marks et al., 2025). We take advantage of the publicly released Gemma Scope SAEs for `Gemma-2-2B` for our experiments (Lieberum et al., 2024).

F.1 Setup

We use the Gemma Scope JumpReLU SAEs of width 16k, one for the residual stream after each of the 26 layers, which gives 425,984 latents in total. For comparison, `Gemma-2-2B` has 239,616 MLP neurons and 234 components. A JumpReLU SAE sets a latent's activation to zero when its pre-activation is below a learned per-latent threshold and passes it through unchanged otherwise. The thresholds were trained with a penalty on L_0, the mean number of active latents per token, using straight-through estimators (Rajamanoharan et al., 2024). At each layer we choose the SAE whose L_0 is closest to 100. On our task prompts, the fraction of unexplained variance ranges from 0.06 at layer 0 to 0.53 at layer 22.

Following Marks et al. (2025), we splice each SAE into the forward pass, so that the residual stream becomes $x = \text{decode}(a) + e$, where a is the vector of latent activations and $e = x - \text{decode}(a)$ is the SAE error term. We compute e on the clean input and hold it fixed, so the model with the SAEs spliced in gives the same outputs as the original model. To ablate a latent, we set its activation in a to zero and leave e unchanged.

Since latent activations are sparse, only latents whose activation differs between the clean and the corrupted input receive a nonzero attribution score, which is about 10,700 to 31,100 latents per example. As a result, we take top-K to be $K\%$ of the latents scored on each example (not the entire set of latents), and we draw the capacity-conserved control from the latents that are scored on at least one training example.

F.2 Consistency and Necessity

In Figures 7 and 8 we show **Reuse**@P and necessity for every task over all values of K and P, where each row is a different granularity, so we can compare feature circuits with the component and neuron circuits from the main text.

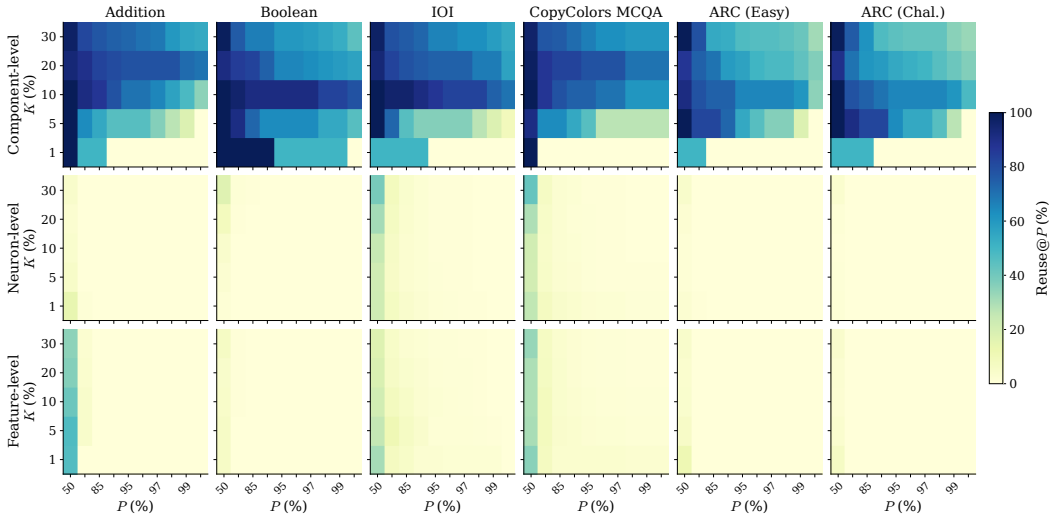


Figure 7: **Reuse@P over K and P** for component, neuron and feature circuits, for `Gemma-2-2B` under EAP-IG.

We find that, for `Gemma-2-2B`, feature circuits behave much like neuron circuits. At K=10% and P=50%, **Reuse**@P is low for most tasks, and the tasks rank in nearly the same order as for neuron circuits, with `IOI` and `CopyColors MCQA` (22 and 29%) well above `Boolean Logic` and the two

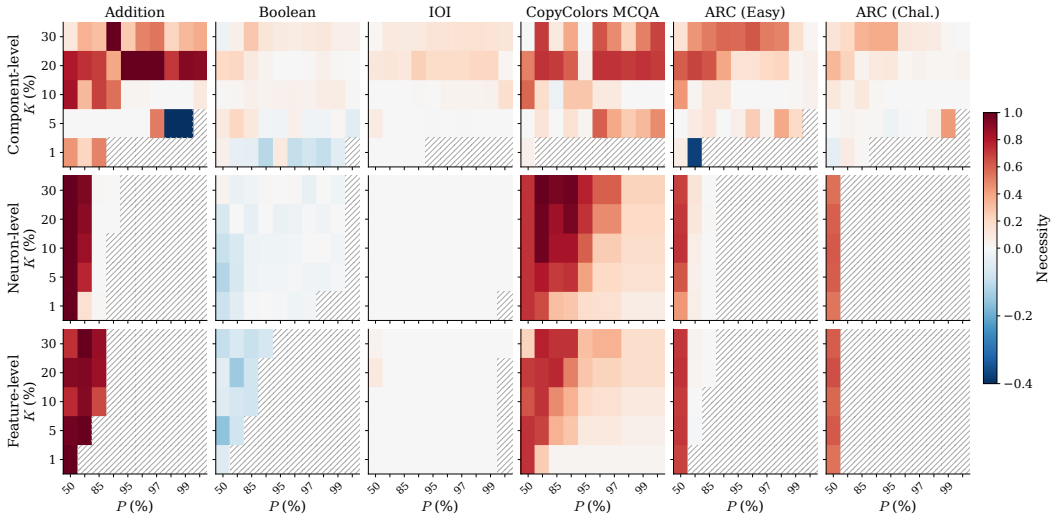


Figure 8: **Necessity over K and P** for component, neuron and feature circuits, for `Gemma-2-2B` under EAP-IG. Hatched cells have an empty task circuit.

`ARC` tasks (2 to 5%). The one large difference is `Addition`, whose **Reuse**@P rises from 4% for neurons to 41% for latents, and this task also accounts for most of the difference in mean **Reuse**@P between the two granularities (17.3% against 9.4%). Both, however, remain far below component circuits, which reach a **Reuse**@P of 91 to 100% on every task. These patterns hold across K, and as with neuron circuits, **Reuse**@P falls quickly as P increases, with no task exceeding 8.4% at K=10% and P=75%.

We also find that the necessity of feature circuits is close to that of neuron circuits on every task, across all values of K. At K=10% and P=50% it ranges from 0.57 to 0.75 for `Addition`, `CopyColors MCQA` and the two `ARC` tasks, compared to 0.61 to 1.03 for neuron circuits, and it is near zero for `Boolean Logic` and `IOI` at all three granularities. `Boolean Logic` has a baseline of 0.51 on a two-way task, so its near-zero necessity is expected. For `IOI`, ablating the 256 latents shared at P=50% leaves accuracy at 1.00, yet ablating each example's own top-10% latents lowers the log-probability of the correct name by 12 to 20 nats. We therefore conclude that the latents shared across `IOI` examples encode the prompt template, while the latents that select the indirect object differ from example to example.

F.3 Specificity

In Figure 9 we plot, for each task, the accuracy drop from ablating its own circuit against the mean drop from ablating the other five tasks' circuits, as in Figure 4. Each accuracy drop is reported as a percentage of the task's baseline accuracy, so that tasks with low baselines can be compared with the other tasks.

We find that feature circuits are also as specific as neuron circuits, and that the two agree closely on each task. At K=10% and P=50%, the mean specificity gap is 28.3% for feature circuits and 29.7% for neuron circuits, and the gaps for individual tasks differ by at most 8 points between the two granularities. For both, ablating a task's own circuit costs `Addition` all of its accuracy and costs `CopyColors MCQA` and the two `ARC` tasks 55 to 72% of theirs, whereas ablating another task's circuit costs these tasks 26 to 31%. Component circuits, in contrast, lie on the diagonal, with a mean gap of -0.2%. `Boolean Logic` and `IOI` show no gap at any granularity, which is consistent with their near-zero necessity.

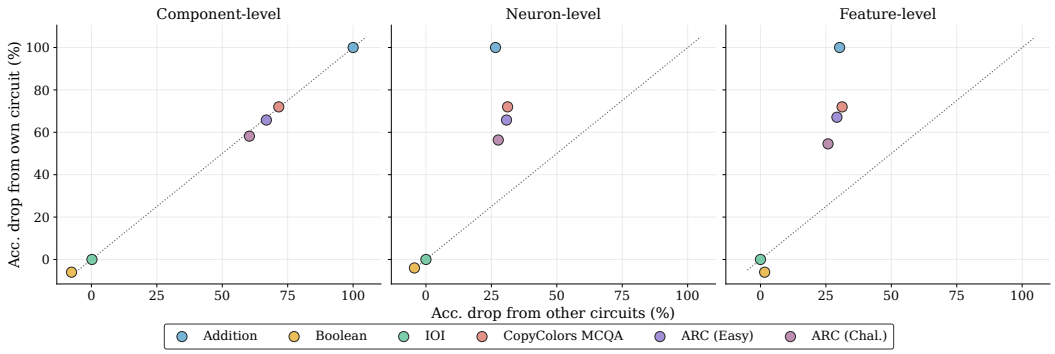


Figure 9: **Accuracy drop from ablating each task's own circuit against the mean drop from ablating the other tasks' circuits**, for `Gemma-2-2B` under EAP-IG at K=10% and P=50%. Drops are measured as a percentage of the task's baseline accuracy, and points above the diagonal indicate a specific circuit.

F.4 Circuit size

Since feature circuits are the same as neuron circuits in terms of necessity and specificity, we next compare how large these circuits are. In Figure 10 we show the mean size of the tasks circuits for neurons and SAE latents, showing the trend across K at P=50% and across P at K=10%.

We observe that feature circuits are roughly ten times smaller than neuron circuits. For example, at K=10% and P=50%, the average circuit has 226 latents against 2,242 neurons, and they stays between 9 and 11 times larger up to P=95%. Across K at P=50%, the ratio grows from 8 at K=1% to 19 at K=30%. Nonetheless, feature circuits reach a similar necessity and specificity to neuron circuits with about a tenth of the units.

F.5 Dictionary size and attribution method

The difference in consistency between SAE latents and neurons could be due to dictionary size. In particular, larger SAEs tend to split a latent into several more specific latents (Bricken et al., 2023), which could lower **Reuse**@P. To test this, we recompute **Reuse**@P and necessity for width-65k SAEs on the three tasks with the highest **Reuse**@P. The width-65k SAEs have 1,703,936 latents in total, about seven times the number of MLP neurons.

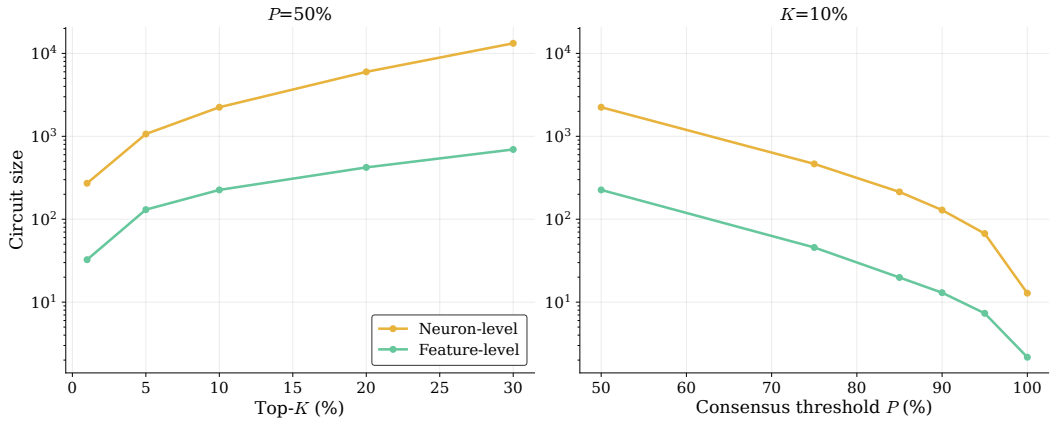


Figure 10: **Mean task circuit size of neuron and feature circuits**, against K at P=50% (left) and against P at K=10% (right), for `Gemma-2-2B` under EAP-IG.

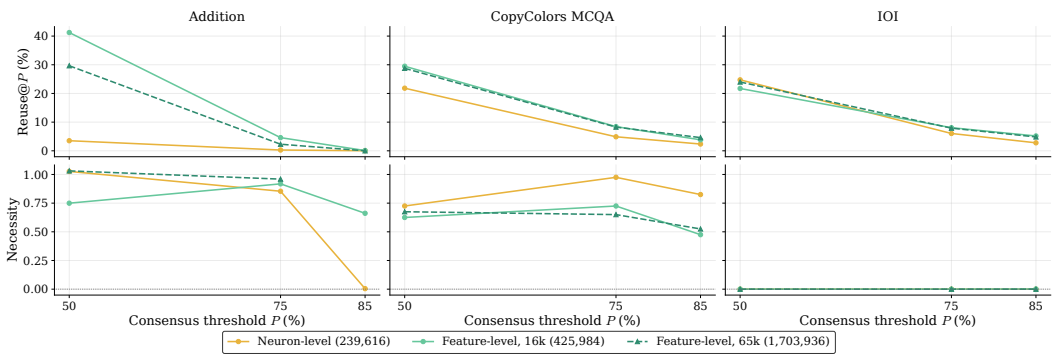


Figure 11: **Dictionary-size control. Reuse@P** and necessity for the width-16k and width-65k SAEs and for neurons, for `Addition`, `CopyColors MCQA`, and `IOI` .

Figure 11 shows that, despite using larger SAEs, **Reuse@P** stays the same for `CopyColors MCQA` and `IOI` and actually drops slightly for `Addition`. Therefore, even at larger dictionary sizes, feature circuits remain more consistent than neuron circuits.

In Figure 12 we compare results between EAP-IG and RelP, at K=10% and P=50%. We find that the two attribution methods largely agree. The correlation between EAP-IG and RelP is r=0.92 for **Reuse@P**, 0.73 for necessity and 0.87 for the specificity gap. Under both methods, component circuits have the highest **Reuse@P**, the lowest necessity and the smallest specificity gap, while neuron and feature circuits are close to each other on all three properties. RelP does, however, find less **Reuse@P** than EAP-IG, especially for component circuits.

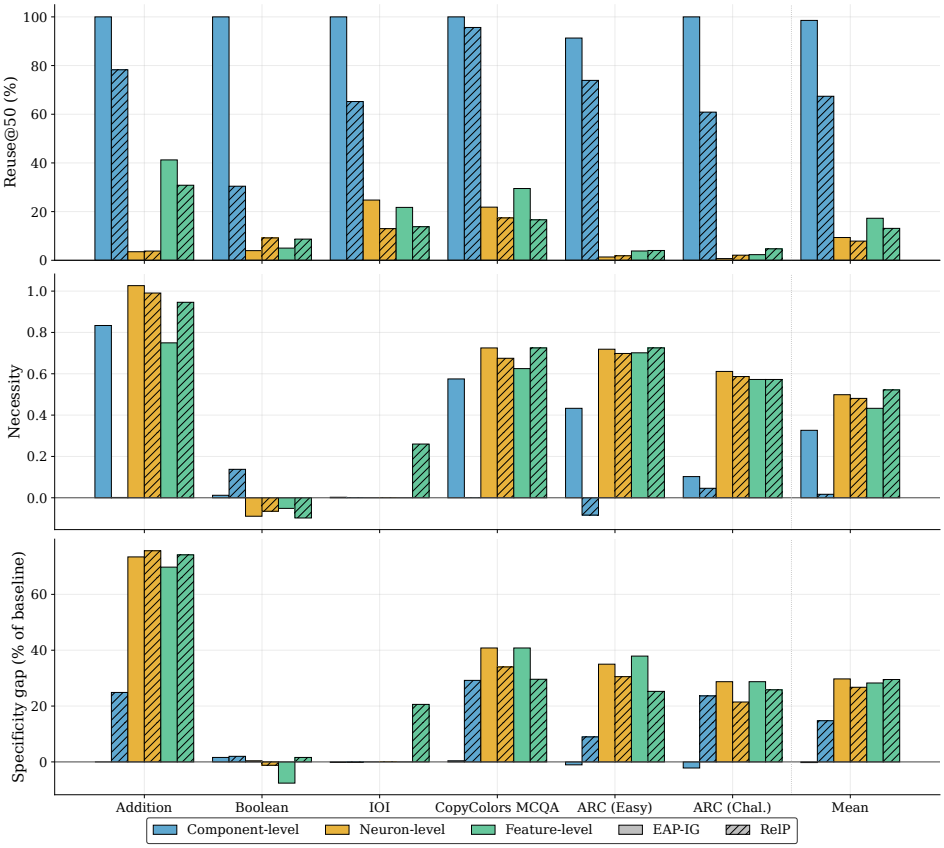


Figure 12: **EAP-IG and RelP** at K=10% and P=50%. Hatched bars are RelP.

G Circuit Size and Composition

G.1 Agreement Between Attribution Methods

In Figure 13 we show, for every model and task, the correlations between EAP-IG and RelP circuits. Specifically, we compute the Spearman correlation in the scores they assign to circuit units and the overlap between the task circuits they produce. The scores agree the most in the two `Llama-3.2-3B` models (Spearman ρ of 0.39 to 0.61) and the least in the `Gemma-2` models (0.13 to 0.42). Agreement in scores does not carry over to the circuits, since even in the `Llama-3.2-3B` models the two methods' component-level task circuits overlap by only 0.15 to 0.45. At the neuron level, only the circuits of `Addition` overlap by more than 0.3 (0.32 to 0.43).

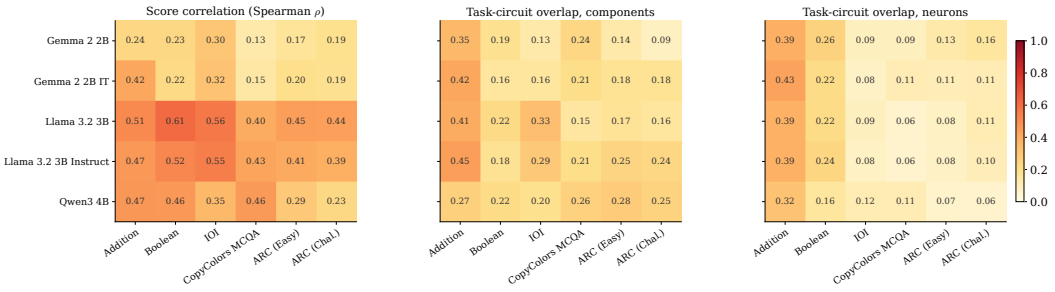


Figure 13: **Agreement between EAP-IG and RelP for each model and task.** *Left:* Spearman correlation between EAP-IG component scores and the magnitude of RelP component scores on the same example, averaged over examples. *Middle and right:* Jaccard overlap between the EAP-IG and RelP task circuits at K=10% and P=50%, at the component and neuron level.

G.2 Circuit Size

Figure 14 reports the absolute size of the within-task circuits across top-K and consensus thresholds P. The two granularities differ substantially in magnitude. At K=10% and P=50%, component-level task circuits typically contain tens of attention heads and MLP blocks, whereas neuron-level task circuits contain hundreds to thousands of neurons. The size of EAP-IG component-level circuits is set mainly by K, as raising P from 50% to 100% shrinks them by a factor of 2.6 to 3.4 for $K \geq 5\%$, whereas RelP component-level circuits and neuron-level circuits of both methods shrink much faster with P (neuron-level circuits by a factor of 5 to 10 from P=50% to 75% alone).

G.3 Component Types

Tables 6 and 7 report the attention head and MLP block percentages of each task circuit across all models, tasks, and top-K, at P=50%. The neuron granularity has no counterpart, since it decomposes the MLPs alone.

G.4 Component Counts by Model

Tables 8–12 report the mean number of attention heads and MLP blocks in each circuit for all models and tasks under EAP-IG, extending the summary in the main text to the full set of models.

G.5 Cross-Task Circuit Decomposition Sizes

The analysis above considers the size of the within-task circuit S_P. For the cross-task experiments, we additionally partition pairs of task circuits into their shared core, task-only set, and partner-only set, as described in §5. Tables 13–16 give the mean number of components in each partition for all models and tasks, averaged over the five partner tasks, at P=50%.

The fraction of the union contained in the shared core differs across attribution methods. Under EAP-IG it is 38–61% at K=5% and 53–62% at K=30%, with the largest values at K=10%, where the

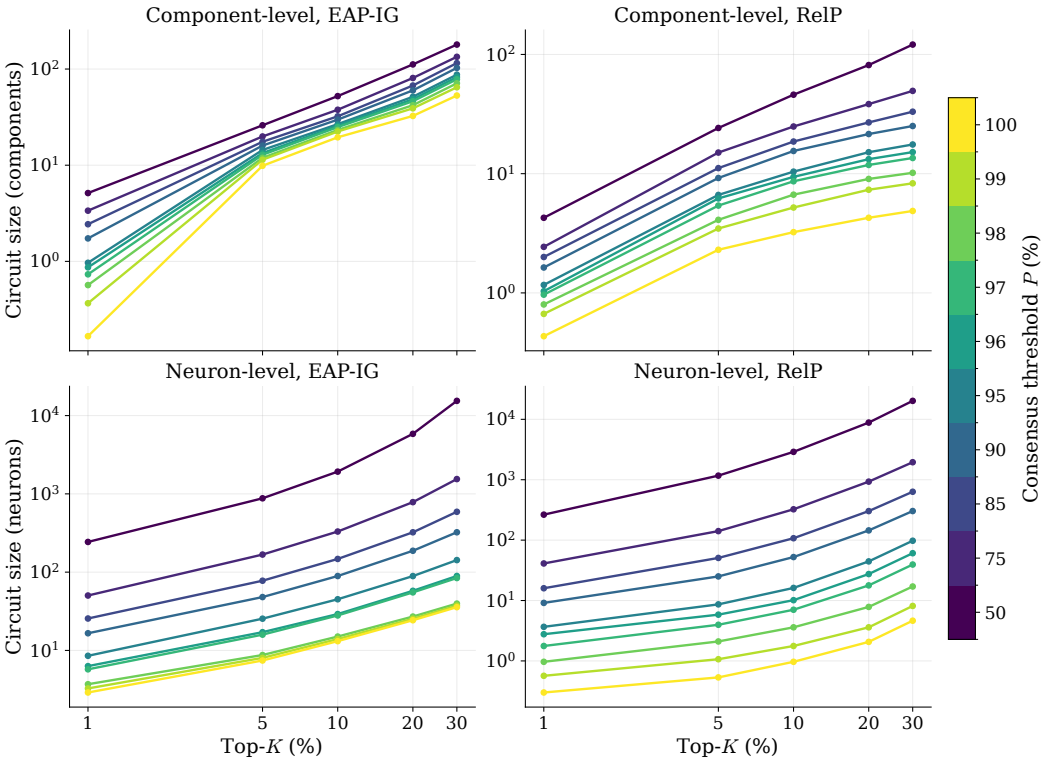


Figure 14: **Task circuit size across K and P.** Number of components (top) and neurons (bottom) in the task circuit across top-K, one line per consensus threshold P, for EAP-IG (left) and RelP (right), averaged over models and tasks with empty circuits counted as zero. Both axes are logarithmic.

Gemma models reach 83% and 88% compared with 45–49% for the Llama and Qwen families. Under RelP the shared core is smaller, between 11% and 31% at every K and in every model, consistent with its lower cross-task overlap.

K	Model	Addition	ARC (Chal.)	ARC (Easy)	Boolean	IOI	CopyColors MCQA
1%	`Gemma 2 2B`	0/100	0/100	0/100	0/100	0/100	0/100
	`Gemma 2 2B Instruct`	0/100	0/100	0/100	–	–	0/100
	`Llama-3.2-3B`	33/67	0/100	0/100	0/100	40/60	33/67
	`Llama-3.2-3B Instruct`	17/83	0/100	0/100	0/100	29/71	17/83
	`Qwen3-4B`	0/100	0/100	0/100	0/100	9/91	45/55
5%	`Gemma 2 2B`	0/100	0/100	0/100	0/100	0/100	0/100
	`Gemma 2 2B Instruct`	0/100	0/100	0/100	0/100	0/100	0/100
	`Llama-3.2-3B`	18/82	38/62	36/64	42/58	34/66	50/50
	`Llama-3.2-3B Instruct`	15/85	33/67	37/63	44/56	41/59	45/55
	`Qwen3-4B`	38/62	48/52	39/61	47/53	40/60	42/58
10%	`Gemma 2 2B`	0/100	17/83	4/96	5/95	5/95	8/92
	`Gemma 2 2B Instruct`	0/100	0/100	0/100	5/95	5/95	0/100
	`Llama-3.2-3B`	51/49	71/29	66/34	74/26	65/35	67/33
	`Llama-3.2-3B Instruct`	53/47	65/35	62/38	70/30	65/35	66/34
	`Qwen3-4B`	65/35	67/33	63/37	67/33	71/29	67/33
20%	`Gemma 2 2B`	40/60	40/60	35/65	48/52	42/58	38/62
	`Gemma 2 2B Instruct`	43/57	35/65	37/63	48/52	47/53	42/58
	`Llama-3.2-3B`	78/22	84/16	82/18	85/15	79/21	80/20
	`Llama-3.2-3B Instruct`	79/21	81/19	78/22	85/15	80/20	80/20
	`Qwen3-4B`	84/16	83/17	82/18	84/16	83/17	84/16
30%	`Gemma 2 2B`	62/38	64/36	63/37	64/36	63/37	63/37
	`Gemma 2 2B Instruct`	60/40	62/38	61/39	64/36	60/40	61/39
	`Llama-3.2-3B`	86/14	89/11	88/12	89/11	87/13	87/13
	`Llama-3.2-3B Instruct`	86/14	87/13	88/12	88/12	86/14	86/14
	`Qwen3-4B`	90/10	90/10	90/10	90/10	89/11	89/11

Table 6: Attention head / MLP block percentage of the shared set under EAP-IG, at P=50%. A dash marks an empty shared set.

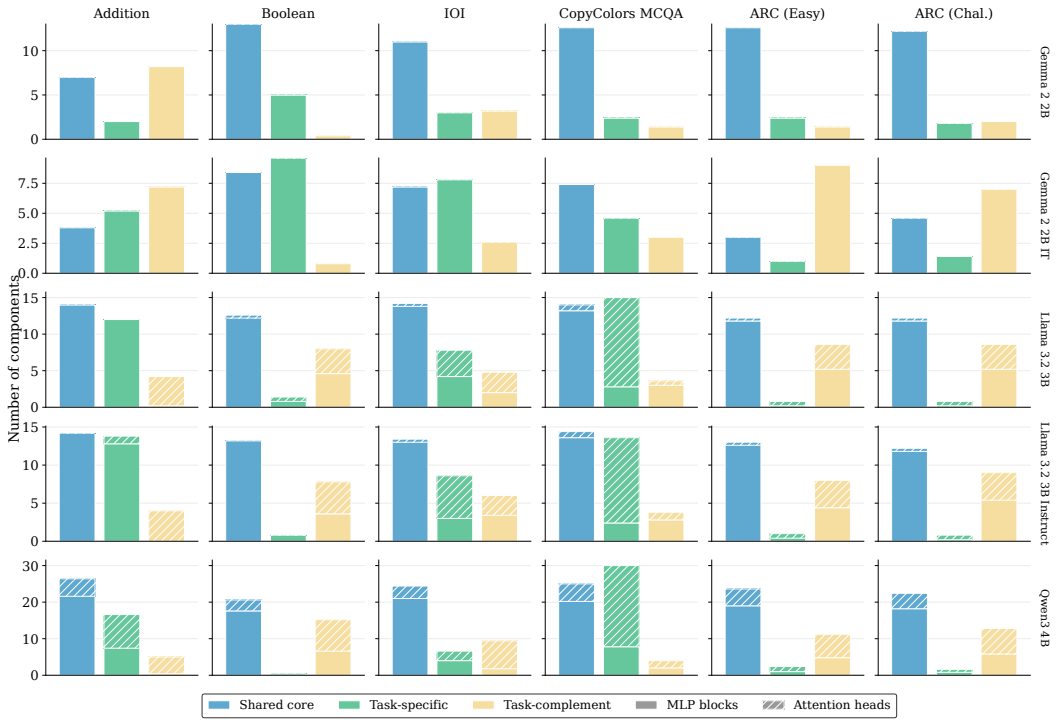


Figure 15: **MLP vs. attention head composition of circuit decompositions at K=10%.** Each group of three bars shows the mean number of MLP blocks (solid) and attention heads (hatched) in the shared core, task-specific, and task-complement sets. MLP blocks account for the vast majority of the shared core across all models and tasks, while attention heads appear primarily in the task-specific and task-complement sets at larger circuit sizes.

K	Model	Addition	ARC (Chal.)	ARC (Easy)	Boolean	IOI	CopyColors MCQA
1%	`Gemma 2 2B`	0/100	100/0	100/0	–	100/0	100/0
	`Gemma 2 2B Instruct`	0/100	100/0	100/0	–	100/0	100/0
	`Llama-3.2-3B`	0/100	75/25	75/25	67/33	100/0	83/17
	`Llama-3.2-3B Instruct`	0/100	75/25	75/25	100/0	100/0	83/17
	`Qwen3-4B`	18/82	58/42	55/45	50/50	100/0	60/40
5%	`Gemma 2 2B`	12/88	50/50	40/60	62/38	100/0	55/45
	`Gemma 2 2B Instruct`	22/78	55/45	55/45	50/50	100/0	55/45
	`Llama-3.2-3B`	33/67	71/29	68/32	60/40	62/38	75/25
	`Llama-3.2-3B Instruct`	32/68	68/32	69/31	54/46	72/28	70/30
	`Qwen3-4B`	57/43	58/42	57/43	69/31	69/31	72/28
10%	`Gemma 2 2B`	39/61	47/53	53/47	68/32	81/19	55/45
	`Gemma 2 2B Instruct`	38/62	56/44	59/41	68/32	73/27	57/43
	`Llama-3.2-3B`	47/53	59/41	64/36	67/33	60/40	82/18
	`Llama-3.2-3B Instruct`	44/56	54/46	55/45	62/38	63/37	75/25
	`Qwen3-4B`	66/34	61/39	65/35	74/26	74/26	82/18
20%	`Gemma 2 2B`	52/48	52/48	58/42	73/27	81/19	62/38
	`Gemma 2 2B Instruct`	53/47	56/44	64/36	69/31	77/23	67/33
	`Llama-3.2-3B`	58/42	67/33	70/30	81/19	78/22	86/14
	`Llama-3.2-3B Instruct`	56/44	63/37	67/33	74/26	78/22	84/16
	`Qwen3-4B`	75/25	78/22	79/21	85/15	81/19	88/12
30%	`Gemma 2 2B`	59/41	57/43	65/35	76/24	85/15	69/31
	`Gemma 2 2B Instruct`	62/38	62/38	64/36	76/24	78/22	72/28
	`Llama-3.2-3B`	70/30	77/23	81/19	85/15	84/16	91/9
	`Llama-3.2-3B Instruct`	69/31	76/24	78/22	83/17	86/14	89/11
	`Qwen3-4B`	81/19	85/15	87/13	89/11	86/14	91/9

Table 7: Attention head / MLP block percentage of the shared set under RelP, at P=50%. A dash marks an empty shared set.

Table 8: Gemma 2 2B: mean attention heads / MLPs in the top-$K\%$ circuit.

Task	1%	10%	20%	30%
Addition	0.0 / 3.0	0.3 / 23.7	21.2 / 25.8	45.1 / 25.9
ARC (Chal.)	0.0 / 3.0	4.0 / 20.0	23.9 / 23.1	46.6 / 24.4
ARC (Easy)	0.0 / 3.0	2.7 / 21.3	22.6 / 24.4	45.8 / 25.2
Boolean	0.0 / 3.0	3.1 / 20.9	24.2 / 22.8	47.2 / 23.8
IOI	0.0 / 3.0	1.8 / 22.2	21.9 / 25.1	45.4 / 25.6
CopyColors MCQA	0.0 / 3.0	1.8 / 22.2	21.2 / 25.8	45.1 / 25.9

Table 9: Gemma 2 2B Instruct: mean attention heads / MLPs in the top-$K\%$ circuit.

Task	1%	10%	20%	30%
Addition	0.0 / 3.0	0.1 / 23.9	21.0 / 26.0	45.0 / 26.0
ARC (Chal.)	0.0 / 3.0	2.5 / 21.5	22.7 / 24.3	45.9 / 25.1
ARC (Easy)	0.0 / 3.0	1.5 / 22.5	21.8 / 25.2	45.4 / 25.6
Boolean	0.0 / 3.0	2.7 / 21.3	23.8 / 23.2	47.0 / 24.0
IOI	0.0 / 3.0	2.0 / 22.0	22.0 / 25.0	45.4 / 25.6
CopyColors MCQA	0.0 / 3.0	1.4 / 22.6	21.2 / 25.8	45.0 / 25.9

Table 10: Llama-3.2-3B: mean attention heads / MLPs in the top-$K\%$ circuit.

Task	1%	10%	20%	30%
Addition	1.1 / 5.9	42.0 / 28.0	112.0 / 28.0	182.0 / 28.0
ARC (Chal.)	0.4 / 6.6	52.9 / 17.1	119.9 / 20.1	187.3 / 22.7
ARC (Easy)	0.9 / 6.1	50.9 / 19.1	117.7 / 22.3	185.6 / 24.4
Boolean	0.7 / 6.3	52.6 / 17.4	119.5 / 20.5	186.6 / 23.4
IOI	2.6 / 4.4	48.2 / 21.8	114.2 / 25.8	182.9 / 27.1
CopyColors MCQA	2.1 / 4.9	49.6 / 20.4	114.8 / 25.2	183.0 / 27.0

Table 11: Llama-3.2-3B Instruct: mean attention heads / MLPs in the top-$K\%$ circuit.

Task	1%	10%	20%	30%
Addition	1.2 / 5.8	42.0 / 28.0	112.0 / 28.0	182.0 / 28.0
ARC (Chal.)	0.2 / 6.8	50.4 / 19.6	117.0 / 23.0	184.8 / 25.2
ARC (Easy)	0.5 / 6.5	48.7 / 21.3	115.6 / 24.4	184.0 / 26.0
Boolean	0.7 / 6.3	52.7 / 17.3	119.1 / 20.9	185.8 / 24.2
IOI	2.7 / 4.3	48.4 / 21.6	114.1 / 25.9	182.7 / 27.3
CopyColors MCQA	1.4 / 5.7	48.6 / 21.4	113.4 / 26.6	182.6 / 27.4

Table 12: Qwen3-4B: mean attention heads / MLPs in the top-$K\%$ circuit.

Task	1%	10%	20%	30%
Addition	1.0 / 11.0	83.8 / 35.1	202.5 / 35.5	321.4 / 35.6
ARC (Chal.)	0.3 / 11.7	89.4 / 29.6	205.8 / 32.2	323.4 / 33.6
ARC (Easy)	0.8 / 11.2	86.5 / 32.5	203.8 / 34.2	322.0 / 35.0
Boolean	0.4 / 11.6	88.5 / 30.5	204.8 / 33.2	322.6 / 34.4
IOI	0.8 / 11.2	88.7 / 30.3	204.0 / 34.0	321.6 / 35.4
CopyColors MCQA	3.6 / 8.4	84.2 / 34.8	202.4 / 35.6	321.3 / 35.7

K	Model	Addition	ARC (Chal.)	ARC (Easy)	Boolean	IOI	CopyColors MCQA
1%	`Gemma 2 2B`	0 / 2 / 1	1 / 0 / 1	1 / 0 / 1	1 / 0 / 1	1 / 0 / 1	1 / 1 / 0
	`Gemma 2 2B Instruct`	0 / 2 / 1	0 / 1 / 1	0 / 1 / 1	0 / 0 / 1	0 / 0 / 1	0 / 1 / 1
	`Llama-3.2-3B`	1 / 5 / 5	4 / 3 / 2	3 / 3 / 3	3 / 4 / 3	2 / 3 / 4	2 / 4 / 4
	`Llama-3.2-3B Instruct`	1 / 5 / 5	4 / 3 / 3	4 / 3 / 3	3 / 4 / 4	3 / 4 / 3	2 / 4 / 4
	`Qwen3-4B`	8 / 3 / 3	8 / 2 / 3	8 / 3 / 2	7 / 3 / 3	8 / 3 / 3	4 / 7 / 7
5%	`Gemma 2 2B`	4 / 7 / 7	8 / 3 / 3	8 / 3 / 3	9 / 3 / 2	6 / 4 / 5	8 / 3 / 3
	`Gemma 2 2B Instruct`	4 / 6 / 7	7 / 4 / 3	7 / 3 / 4	8 / 5 / 3	7 / 3 / 4	7 / 3 / 4
	`Llama-3.2-3B`	16 / 18 / 11	17 / 7 / 12	18 / 7 / 11	17 / 9 / 12	16 / 13 / 12	19 / 13 / 8
	`Llama-3.2-3B Instruct`	17 / 16 / 11	19 / 5 / 11	20 / 7 / 9	18 / 9 / 11	18 / 14 / 10	20 / 9 / 9
	`Qwen3-4B`	32 / 23 / 18	34 / 12 / 19	38 / 13 / 13	33 / 16 / 19	32 / 15 / 20	38 / 22 / 11
10%	`Gemma 2 2B`	20 / 4 / 3	19 / 4 / 4	20 / 3 / 3	20 / 2 / 3	20 / 2 / 3	21 / 4 / 2
	`Gemma 2 2B Instruct`	20 / 3 / 3	20 / 2 / 2	21 / 2 / 2	20 / 2 / 3	20 / 2 / 3	21 / 3 / 2
	`Llama-3.2-3B`	26 / 31 / 32	32 / 23 / 27	34 / 22 / 25	29 / 33 / 29	28 / 32 / 30	32 / 28 / 26
	`Llama-3.2-3B Instruct`	27 / 33 / 33	36 / 21 / 24	37 / 21 / 23	31 / 25 / 29	30 / 32 / 29	35 / 29 / 23
	`Qwen3-4B`	49 / 51 / 48	63 / 31 / 35	66 / 29 / 33	58 / 37 / 40	54 / 44 / 44	61 / 44 / 36
20%	`Gemma 2 2B`	32 / 11 / 10	30 / 10 / 12	32 / 8 / 10	29 / 13 / 13	30 / 15 / 11	32 / 10 / 10
	`Gemma 2 2B Instruct`	33 / 13 / 10	32 / 8 / 12	33 / 8 / 11	31 / 13 / 12	31 / 16 / 12	34 / 11 / 9
	`Llama-3.2-3B`	68 / 59 / 64	80 / 47 / 51	86 / 42 / 46	77 / 61 / 53	70 / 59 / 61	83 / 53 / 47
	`Llama-3.2-3B Instruct`	70 / 61 / 57	79 / 36 / 52	85 / 40 / 43	73 / 57 / 54	68 / 62 / 59	83 / 53 / 43
	`Qwen3-4B`	123 / 100 / 90	135 / 69 / 81	139 / 65 / 77	129 / 91 / 83	124 / 87 / 91	122 / 100 / 90
30%	`Gemma 2 2B`	51 / 17 / 19	55 / 18 / 14	54 / 16 / 16	51 / 16 / 19	48 / 23 / 22	54 / 17 / 16
	`Gemma 2 2B Instruct`	48 / 17 / 19	51 / 18 / 15	50 / 16 / 16	51 / 16 / 16	46 / 19 / 20	52 / 15 / 14
	`Llama-3.2-3B`	139 / 68 / 70	152 / 57 / 56	157 / 52 / 51	145 / 67 / 63	137 / 65 / 73	154 / 59 / 54
	`Llama-3.2-3B Instruct`	129 / 72 / 81	159 / 55 / 48	166 / 50 / 41	153 / 58 / 55	144 / 58 / 66	154 / 53 / 55
	`Qwen3-4B`	221 / 126 / 123	250 / 99 / 94	253 / 96 / 90	236 / 111 / 108	224 / 113 / 123	215 / 124 / 131

Table 13: Mean shared core / task-only / partner-only sizes over attention heads and MLP blocks under EAP-IG, at P=50%, averaged over the five partner tasks.

K	Model	Addition	ARC (Chal.)	ARC (Easy)	Boolean	IOI	CopyColors MCQA
1%	Gemma 2 2B	0/2/1	0/1/1	0/1/1	0/0/1	0/2/1	0/1/1
	Gemma 2 2B Instruct	0/2/1	0/1/1	0/1/1	0/0/1	0/2/1	0/1/1
	Llama-3.2-3B	1/7/4	2/2/3	2/2/3	1/5/4	0/3/6	2/4/3
	Llama-3.2-3B Instruct	1/6/3	2/2/3	2/2/3	0/2/5	0/3/5	2/4/2
	Qwen3-4B	4/7/5	6/6/3	6/5/3	1/7/9	2/3/9	5/5/4
5%	Gemma 2 2B	2/6/6	5/3/4	5/5/3	1/7/7	1/6/8	5/6/3
	Gemma 2 2B Instruct	3/6/7	6/5/3	6/5/3	2/4/8	2/7/7	6/5/3
	Llama-3.2-3B	12/18/16	14/10/14	17/11/11	11/14/18	13/16/15	17/15/10
	Llama-3.2-3B Instruct	12/19/14	15/7/13	16/10/11	11/15/16	12/13/15	16/14/10
	Qwen3-4B	20/27/28	30/13/19	31/16/17	22/30/25	23/26/25	29/21/19
10%	Gemma 2 2B	8/10/12	10/5/10	12/7/8	8/14/11	8/13/11	13/9/6
	Gemma 2 2B Instruct	9/12/11	12/6/9	12/5/9	8/14/12	9/13/11	14/9/6
	Llama-3.2-3B	23/24/30	29/12/26	30/15/23	25/35/25	30/30/21	28/32/22
	Llama-3.2-3B Instruct	26/24/28	33/15/21	35/18/18	27/28/25	30/27/22	28/25/25
	Qwen3-4B	29/38/59	48/22/40	51/27/36	45/63/35	41/57/42	47/42/37
20%	Gemma 2 2B	16/15/22	22/9/15	23/10/14	19/22/17	16/27/19	25/15/11
	Gemma 2 2B Instruct	18/20/24	27/12/15	27/15/14	21/18/20	20/23/20	28/17/13
	Llama-3.2-3B	39/28/57	46/17/50	51/23/43	46/72/40	47/65/40	50/60/37
	Llama-3.2-3B Instruct	35/28/54	45/18/44	49/27/37	41/52/41	42/63/38	46/59/34
	Qwen3-4B	44/58/122	90/51/68	94/60/61	81/109/67	67/94/87	84/98/65
30%	Gemma 2 2B	24/22/28	29/8/25	33/15/19	28/31/22	26/33/24	34/24/16
	Gemma 2 2B Instruct	29/26/31	40/16/20	41/18/19	33/25/26	30/29/29	41/27/16
	Llama-3.2-3B	52/42/88	67/30/73	73/44/63	69/100/57	66/94/61	72/88/55
	Llama-3.2-3B Instruct	48/41/88	67/32/68	72/41/60	59/85/66	64/97/58	69/96/52
	Qwen3-4B	71/85/178	131/81/107	141/104/90	122/156/103	93/122/144	131/163/90

Table 14: Mean shared core / task-only / partner-only sizes over attention heads and MLP blocks under RelP, at P=50%, averaged over the five partner tasks.

K	Model	Addition	ARC (Chal.)	ARC (Easy)	Boolean	IOI	CopyColors MCQA
1%	Gemma 2 2B	1/347/285	14/23/334	20/53/321	3/329/286	4/222/307	23/740/180
	Gemma 2 2B Instruct	1/405/228	15/25/287	19/48/278	3/262/255	6/248/254	21/499/186
	Llama-3.2-3B	3/301/208	10/18/256	15/41/246	4/291/209	2/128/244	16/531/146
	Llama-3.2-3B Instruct	1/305/183	7/13/235	11/29/227	3/112/220	3/195/203	11/539/124
	Qwen3-4B	1/363/201	17/30/248	20/48/241	7/417/183	6/134/241	20/310/189
5%	Gemma 2 2B	8/618/1098	35/57/1178	56/164/1132	23/1105/983	29/831/1031	86/3145/500
	Gemma 2 2B Instruct	9/751/893	33/54/1004	53/173/956	19/799/871	24/940/837	70/2345/501
	Llama-3.2-3B	22/603/953	32/49/1053	48/123/1019	37/1044/848	27/643/939	75/2801/450
	Llama-3.2-3B Instruct	15/597/810	27/48/905	39/103/879	18/332/859	23/943/730	57/2532/372
	Qwen3-4B	9/836/755	46/81/862	56/153/836	25/1191/665	19/452/820	62/1735/512
10%	Gemma 2 2B	25/826/2413	63/110/2511	112/348/2404	77/2785/1959	76/1748/2168	200/6674/1034
	Gemma 2 2B Instruct	29/1065/2015	74/125/2149	116/359/2051	58/1938/1805	71/2085/1761	174/5218/1010
	Llama-3.2-3B	51/865/2277	74/126/2397	105/279/2329	106/2628/1858	89/1464/2111	200/6567/958
	Llama-3.2-3B Instruct	37/858/1835	57/103/1962	77/207/1918	52/712/1846	81/2099/1534	146/5826/711
	Qwen3-4B	26/1308/1816	92/184/1963	114/349/1903	64/2611/1510	50/1054/1838	144/4551/1027
20%	Gemma 2 2B	99/1409/6911	218/403/6969	368/1091/6651	374/9836/4895	346/4711/5953	747/16953/3024
	Gemma 2 2B Instruct	117/1810/5986	260/475/6081	393/1106/5795	294/7275/4680	344/5854/4904	688/13823/2898
	Llama-3.2-3B	164/1463/7503	391/801/7362	540/1533/7037	533/9951/5362	429/4220/6633	989/18945/3016
	Llama-3.2-3B Instruct	113/1476/5192	221/441/5269	302/812/5098	223/2724/4810	383/5826/3998	622/14971/1882
	Qwen3-4B	134/2543/6989	405/898/6993	513/1648/6714	384/10951/5008	262/4188/6507	706/15661/3679
30%	Gemma 2 2B	409/2984/16349	961/2033/15877	1381/3655/15048	1748/25108/10318	1382/10654/13647	2768/34100/7295
	Gemma 2 2B Instruct	494/4289/14952	1103/2153/14649	1511/3780/13833	1458/20430/10567	1430/12962/12094	2635/29768/7287
	Llama-3.2-3B	529/2517/19590	2411/6785/16478	2737/7545/15935	2602/26052/12396	1686/9211/16863	4197/37369/8218
	Llama-3.2-3B Instruct	351/2516/12869	952/2163/12218	1228/3339/11652	1070/10134/10482	1392/12205/9682	2258/31359/4812
	Qwen3-4B	767/6082/22044	2160/5439/20500	2497/6962/19792	2569/35211/14056	1659/13735/19443	3696/40125/11721

Table 15: Mean shared core / task-only / partner-only sizes over MLP neurons under EAP-IG, at P=50%, averaged over the five partner tasks.

K	Model	Addition	ARC (Chal.)	ARC (Easy)	Boolean	IOI	CopyColors MCQA
1%	`Gemma 2 2B`	3 / 321 / 198	17 / 29 / 240	22 / 58 / 227	4 / 514 / 158	2 / 82 / 247	24 / 252 / 187
	`Gemma 2 2B Instruct`	4 / 364 / 199	27 / 50 / 235	33 / 82 / 221	3 / 393 / 195	0 / 67 / 263	32 / 331 / 172
	`Llama-3.2-3B`	2 / 212 / 238	28 / 51 / 240	32 / 71 / 231	5 / 574 / 163	2 / 38 / 273	34 / 369 / 169
	`Llama-3.2-3B Instruct`	0 / 190 / 141	18 / 35 / 151	22 / 63 / 141	0 / 180 / 143	1 / 74 / 163	21 / 293 / 96
	`Qwen3-4B`	3 / 595 / 456	56 / 119 / 488	62 / 152 / 475	10 / 1243 / 318	8 / 81 / 553	54 / 512 / 412
5%	`Gemma 2 2B`	17 / 579 / 906	42 / 79 / 976	56 / 135 / 949	25 / 2221 / 568	12 / 352 / 957	70 / 1625 / 633
	`Gemma 2 2B Instruct`	22 / 651 / 1074	99 / 185 / 1075	148 / 415 / 970	28 / 1862 / 825	11 / 349 / 1148	155 / 2228 / 599
	`Llama-3.2-3B`	15 / 443 / 1142	80 / 138 / 1125	114 / 288 / 1055	38 / 2475 / 708	16 / 263 / 1177	130 / 2243 / 644
	`Llama-3.2-3B Instruct`	7 / 433 / 743	63 / 120 / 738	81 / 238 / 693	11 / 620 / 700	14 / 326 / 755	81 / 2193 / 301
	`Qwen3-4B`	44 / 1106 / 2381	243 / 476 / 2268	290 / 755 / 2156	104 / 5628 / 1404	41 / 396 / 2527	297 / 3894 / 1519
10%	`Gemma 2 2B`	43 / 894 / 2372	83 / 156 / 2473	131 / 391 / 2368	94 / 5229 / 1445	43 / 867 / 2378	201 / 4883 / 1386
	`Gemma 2 2B Instruct`	60 / 1028 / 3004	290 / 508 / 2832	456 / 1279 / 2479	123 / 4610 / 2212	46 / 936 / 3039	506 / 6566 / 1361
	`Llama-3.2-3B`	45 / 653 / 2927	186 / 299 / 2828	286 / 733 / 2622	137 / 5837 / 1779	56 / 727 / 2899	357 / 6241 / 1434
	`Llama-3.2-3B Instruct`	27 / 689 / 1980	148 / 286 / 1916	217 / 685 / 1753	46 / 1499 / 1796	44 / 856 / 1927	234 / 6023 / 665
	`Qwen3-4B`	119 / 1533 / 5769	661 / 1241 / 5176	812 / 2049 / 4834	378 / 11470 / 3470	90 / 924 / 5925	910 / 10904 / 2946
20%	`Gemma 2 2B`	195 / 2083 / 7441	319 / 616 / 7587	610 / 1875 / 6985	585 / 14099 / 4570	228 / 2779 / 7263	1041 / 16031 / 3637
	`Gemma 2 2B Instruct`	364 / 3220 / 9700	1393 / 2373 / 8635	2026 / 4945 / 7360	845 / 12594 / 7249	264 / 3053 / 9853	2378 / 20450 / 3837
	`Llama-3.2-3B`	180 / 1333 / 8910	763 / 1213 / 8235	1171 / 2811 / 7426	725 / 15531 / 5417	265 / 2427 / 8590	1583 / 18963 / 3701
	`Llama-3.2-3B Instruct`	146 / 1476 / 6749	639 / 1203 / 6212	926 / 2770 / 5554	285 / 5173 / 5843	209 / 2901 / 6388	1099 / 19270 / 2047
	`Qwen3-4B`	526 / 3235 / 16249	2739 / 4593 / 13322	3348 / 7390 / 12031	1985 / 24765 / 10193	385 / 2885 / 16489	3944 / 31842 / 6426
30%	`Gemma 2 2B`	1005 / 6381 / 16477	1588 / 2725 / 16509	2545 / 6283 / 14649	2385 / 27678 / 10561	1072 / 7664 / 16140	3918 / 31553 / 7947
	`Gemma 2 2B Instruct`	1866 / 8837 / 20608	4774 / 7394 / 17407	6067 / 11556 / 15022	3383 / 25327 / 15489	1375 / 8175 / 21329	7217 / 37100 / 8534
	`Llama-3.2-3B`	742 / 3617 / 19538	2840 / 4385 / 16867	3927 / 8330 / 14774	2832 / 29932 / 11768	1090 / 6658 / 18512	5246 / 36161 / 7625
	`Llama-3.2-3B Instruct`	639 / 3992 / 16163	2335 / 4241 / 14078	3150 / 8042 / 12341	1424 / 13800 / 13260	961 / 8171 / 14941	3904 / 37984 / 5447
	`Qwen3-4B`	2699 / 9784 / 33677	8828 / 13432 / 25593	10171 / 18371 / 22993	6970 / 44046 / 21700	1912 / 9238 / 34731	11740 / 57173 / 13350

Table 16: Mean shared core / task-only / partner-only sizes over MLP neurons under RelP, at P=50%, averaged over the five partner tasks.

G.6 Layer Distribution

Figure 16 expands Figure 3 into thirty model–task cells, shown for all four combinations of attribution method and granularity. The ordering described in the main text appears in almost every cell: component-level EAP-IG is generally earliest, while component-level RelP is latest. For EAP-IG, the neuron-level circuit is earlier than the component-level circuit in only three of the thirty cells, including `Addition` for both Llama models. The Gemma models show the least variation across depth. Their component-level EAP-IG circuits differ from the uniform diagonal by a mean of 0.15 and 0.11 across tasks, compared with 0.28–0.31 for the Llama and Qwen models. These Gemma circuits consist of 93.5% and 98.5% MLP blocks, respectively. `Gemma 2 2B` contains only 26 such blocks, so a $K = 10\%$ circuit selects roughly one MLP block per unit of depth across tasks.

Figure 17 shows the same quantity as a function of circuit size top-K rather than of method and granularity, under EAP-IG at the component granularity. In every model, the CDF at K=1% is shifted toward earlier layers than at larger K, indicating that the highest-attribution components are concentrated in early-to-middle layers. The exception is `Addition` in the two `Gemma-2` models, whose K=1% circuit lies mostly in the last few layers. At larger K, the distribution becomes more uniform across layers in all models.

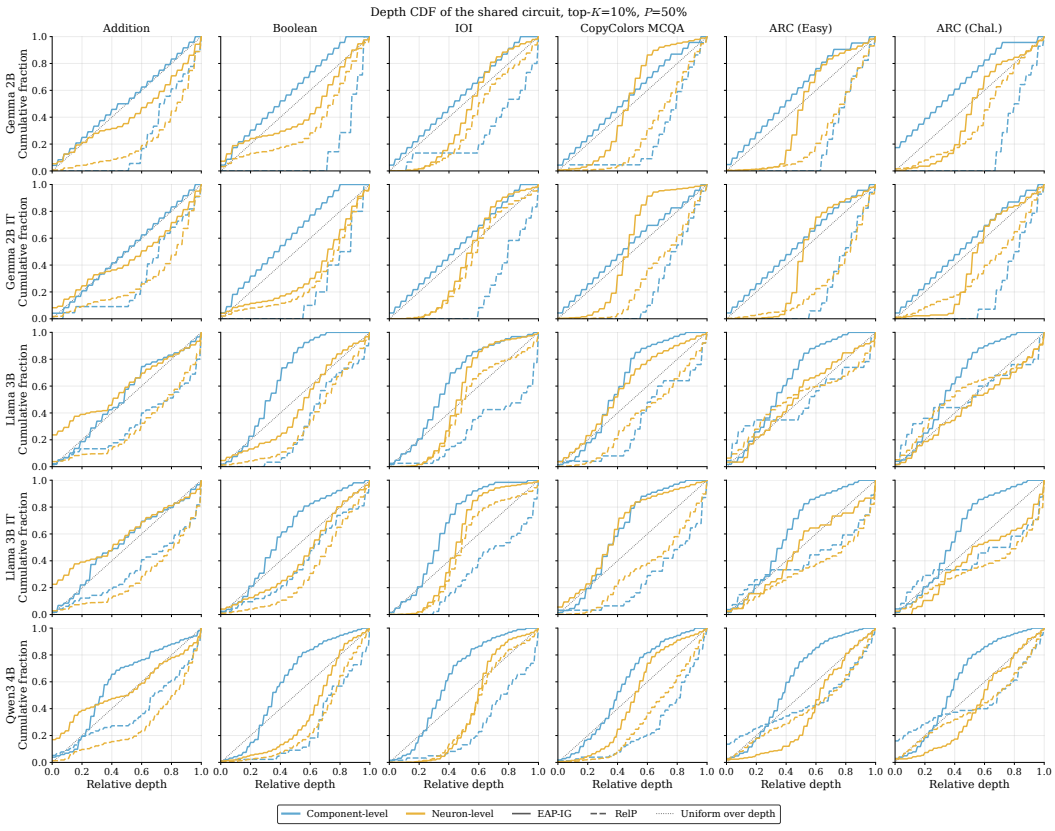


Figure 16: **Depth CDF of the task circuit for every model and task, at K=10% and P=50%.** Each panel gives the cumulative fraction of the task circuit at or below a given relative depth, for all four combinations of attribution method and granularity. The dotted diagonal is a circuit spread uniformly over depth.

H Selective Ablation Across K

Figures 18 and 19 show the selective ablation results for $K \in \{1, 5, 10, 20, 30\}\%$, complementing the K=10% results in the main text.

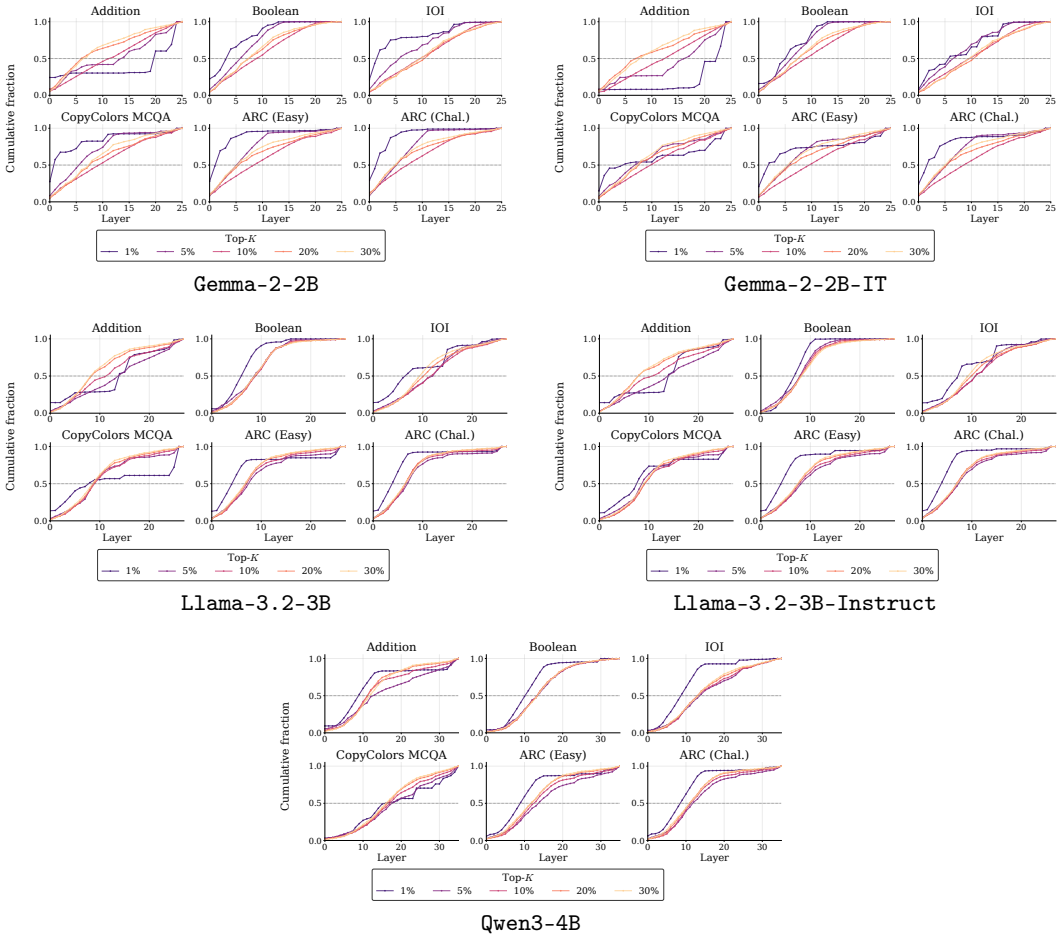


Figure 17: **Cumulative layer distribution of circuit components.** Each line shows the cumulative fraction of components in the top-$K\%$ circuit at or below a given layer. At small K, the CDF is shifted left (toward earlier layers) in every model, except for `Addition` in the `Gemma-2` models.

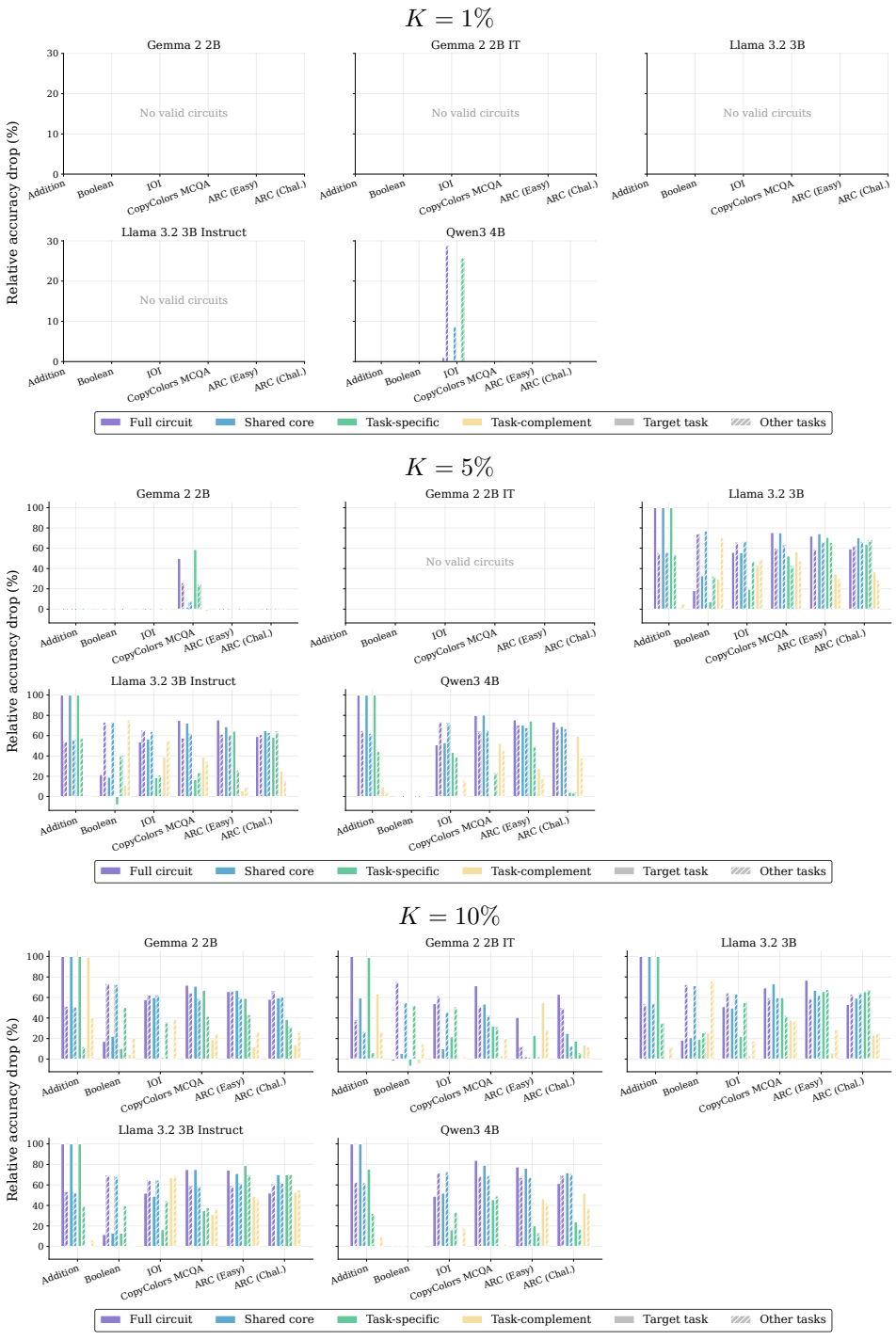


Figure 18: Selective ablation: relative accuracy drop by component set for $K \in \{1, 5, 10\}\%$.

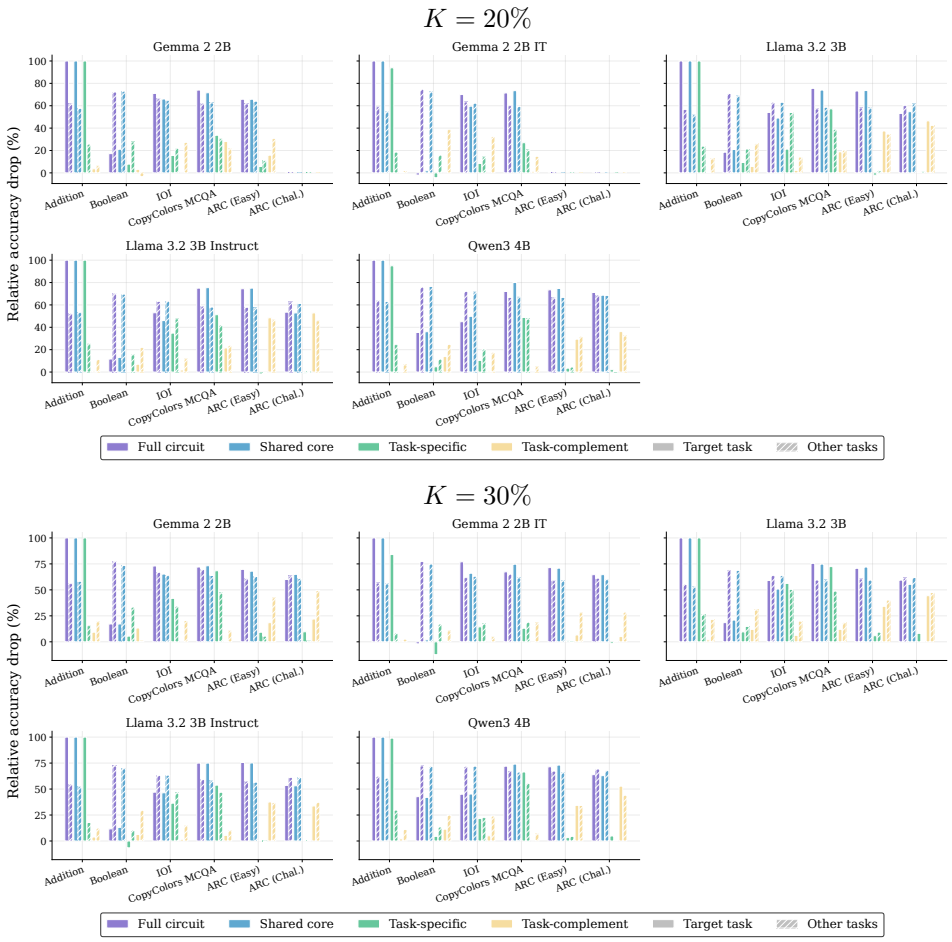


Figure 19: Selective ablation: relative accuracy drop by component set for $K \in \{20, 30\}\%$.

I CASE STUDY OF SHARED COMPONENTS IN LLAMA-3.2-3B

To characterize the three attention heads shared by all six tasks in `Llama-3.2-3B`, L4H10, L9H21 and L15H18, we ran 16 prompts from each task through the model and recorded the attention from the final query position, aggregating it by the template role of each key position (Figure 20). Summing over every token of a role and averaging over tasks, the beginning-of-text token receives 0.762, 0.420 and 0.144 of the attention of L4H10, L9H21 and L15H18, against 0.096, 0.389 and 0.737 for the body of the prompt and 0.018, 0.137 and 0.020 for the final token. L4H10 therefore behaves as an attention sink (Xiao et al., 2024). L15H18 sends more of its attention to the body of the prompt than 99.9% of the heads in the model, and in `Addition`, `Boolean Logic` and `IOI` a large share of it goes to the tokens that carry the task, with 34% on the digits in `Addition`, 47% on the literals in `Boolean Logic` and 47% on the names in `IOI`, each above the 97th percentile of all heads. L9H21 divides its attention between the beginning-of-text token and the body of the prompt, and within the body it attends to the answer labels of `CopyColors MCQA` and the two `ARC` tasks more than 89 to 92% of heads do. We restrict this analysis to the component granularity, since the neuron basis contains no attention, and to the EAP-IG circuits studied in §5.1.

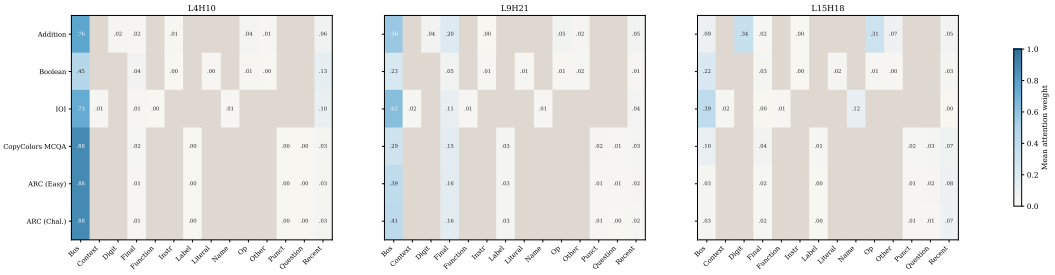


Figure 20: **Attention from the final query position for the three heads shared by all six tasks in `Llama-3.2-3B`.** Rows are tasks, columns are template roles, and each cell is the mean attention weight per token of that role, over 16 prompts. Grey cells mark roles that do not occur in that task. Because these are per-token means rather than totals, they are comparable down a column but should not be summed across a row.

J PRETRAINING DYNAMICS

We track reuse and necessity across ten checkpoints of `OLMo-2-1B`'s ∼4T-token training run, eight from stage 1 and two stage-2 anneals (each ∼51B tokens of curated data with learning-rate decay on top of stage 1), using EAP-IG at the component granularity with 500 examples per task (Figure 21). **Reuse**@95 at K=10% is highest in the first 32B tokens (44–67% across tasks) and then declines, with most of the drop between the 76B and 399B checkpoints. For the rest of stage 1 it lies between 7 and 19% on Addition, 0 and 15% on ARC and CopyColors MCQA, 19 and 26% on IOI, and 0 and 7% on Boolean, whose K=10% task circuit is empty at 4001B. The anneals raise reuse on CopyColors MCQA (to 30–33%) and lower it on IOI (to 7%), the two tasks whose baseline accuracy changes most during training.

Necessity is hard to read for most of stage 1 because baseline accuracy on Addition, ARC and CopyColors MCQA stays near chance, leaving little room for ablation to do measurable damage. During the anneal phase, baselines on Addition (2→37%), ARC (Easy) (27→60%) and CopyColors MCQA (38→98%) jump sharply, likely because anneal mixtures emphasize downstream-task data. With only 272 components, `OLMo-2-1B` is also sensitive to the random control, since removing a handful of random components already costs substantial accuracy. On CopyColors MCQA at the anneals, ablating the K=30% task circuit reduces accuracy from 93–98% to 28–30%, against 38–63% for a capacity-matched random ablation, while at K=10% the control does as much damage as the circuit. The clearest case is `IOI`, whose K=30% circuit is necessary from 9B tokens to the end of stage 1 (necessity of 50 to 70%) and much less so after the anneals (9 to 13%).

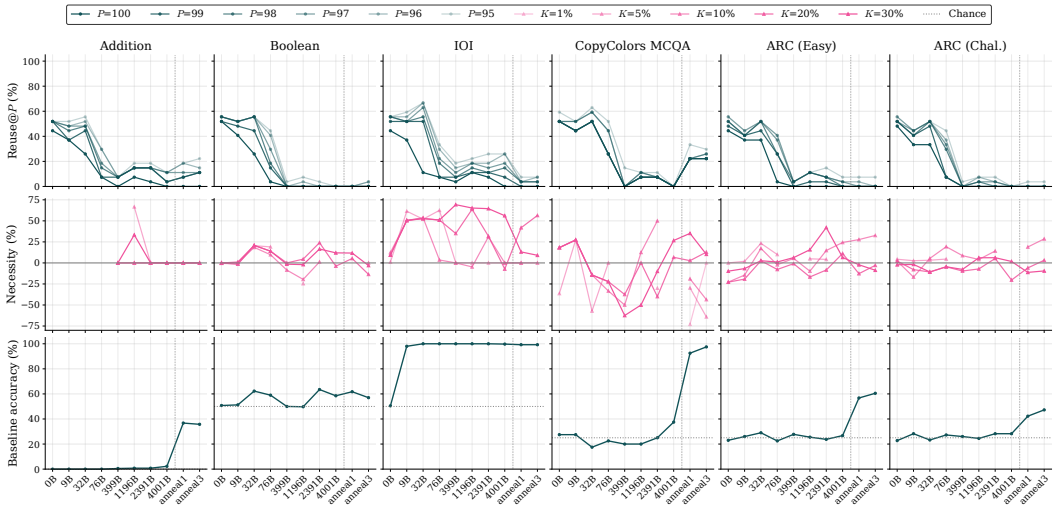


Figure 21: **Circuit reuse, necessity and baseline accuracy across `OLMo-2-1B` pretraining checkpoints.** *Top:* **Reuse**@P at top-K=10% for P from 95% (lightest) to 100% (darkest). *Middle:* necessity of the P=95% task circuit for top-K from 1% (lightest) to 30% (darkest), missing where the baseline accuracy is zero or the circuit is empty. *Bottom:* baseline accuracy, with chance accuracy as a grey dotted line. The vertical dotted line separates the stage-1 checkpoints from the two stage-2 anneals.

K Full Within-Task Results

We report **Reuse**@P across the full sweep of top-K and consensus threshold P for every model and task, separately for each attribution method and granularity. Figure 22 summarizes the effect of varying K and P with the median over models and tasks, while Figures 23–26 show the full $K \times P$ sweeps for each extraction configuration. Figures 27 and 28 give necessity along top-K at P=50% and along P at K=10%, the two sweeps on which we measure it.

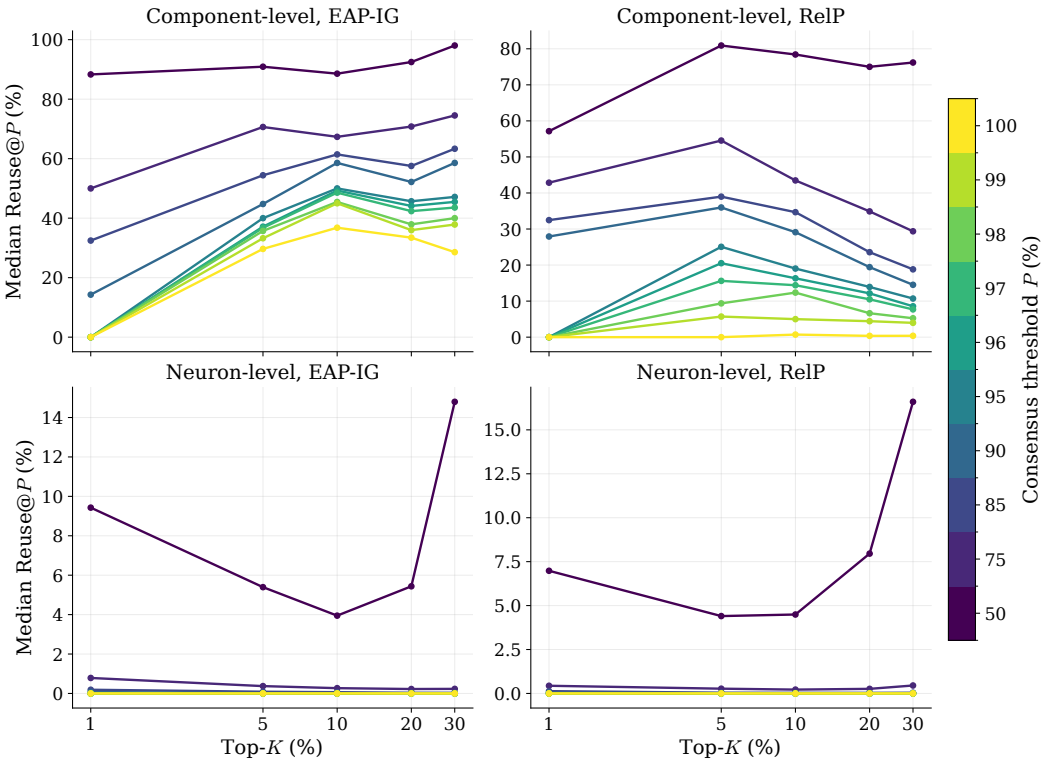


Figure 22: Median within-task **Reuse**@P over models and tasks across top-K, one line per consensus threshold P, for component-level (top) and neuron-level (bottom) circuits under EAP-IG (left) and RelP (right).

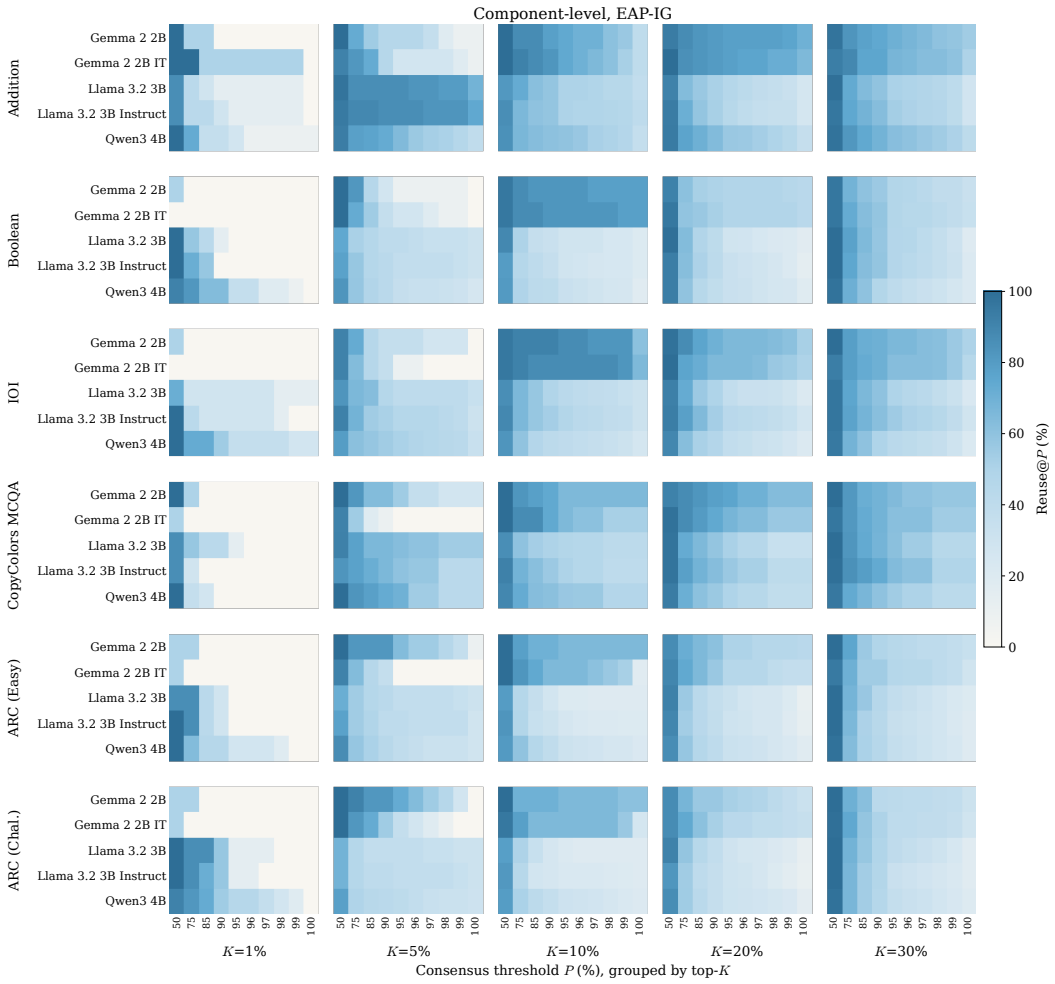


Figure 23: Within-task **Reuse**@P of component-level circuits under EAP-IG over the full grid of top-K and consensus threshold P. Columns are grouped by top-K, and rows are grouped by task, with one row per model. A value of 0 includes settings where the task circuit is empty.

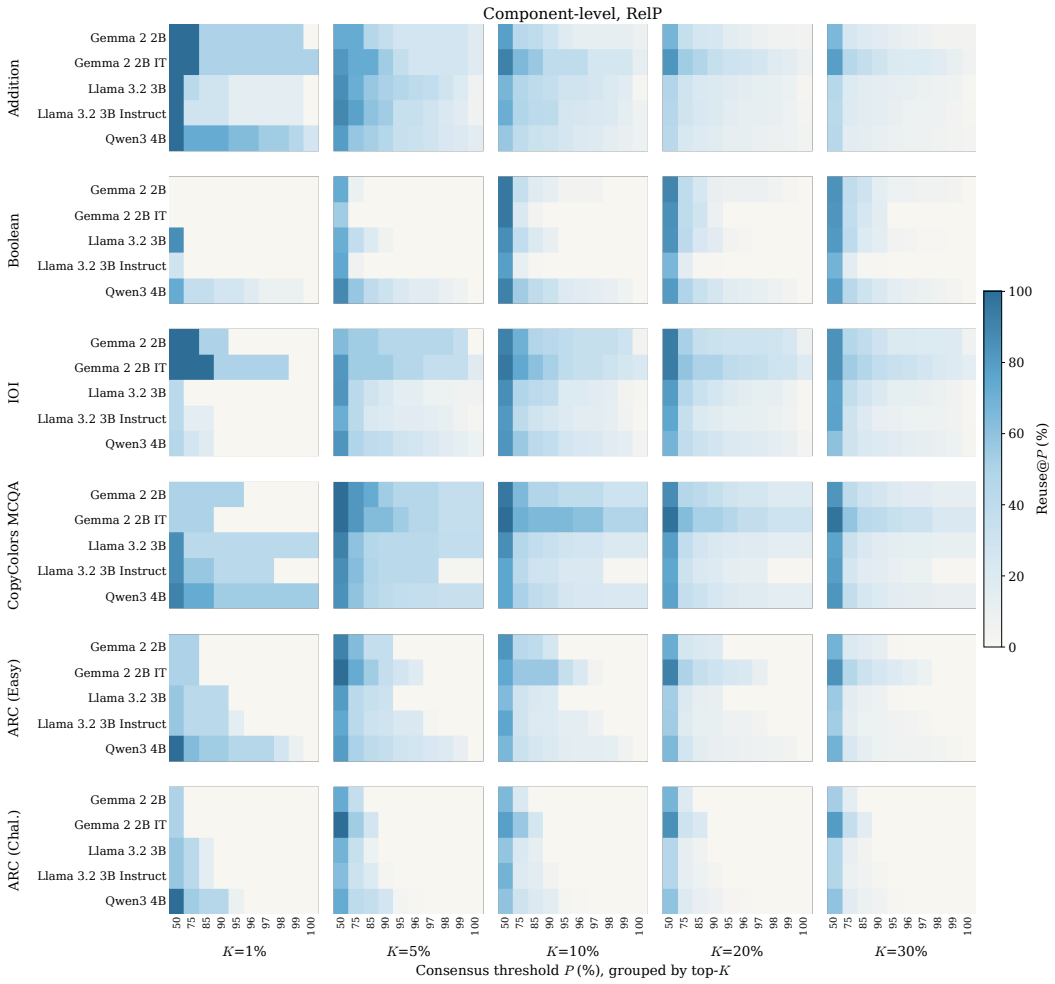


Figure 24: Within-task **Reuse**@P of component-level circuits under RelP over the full grid of top-K and consensus threshold P. Columns are grouped by top-K, and rows are grouped by task, with one row per model. A value of 0 includes settings where the task circuit is empty.

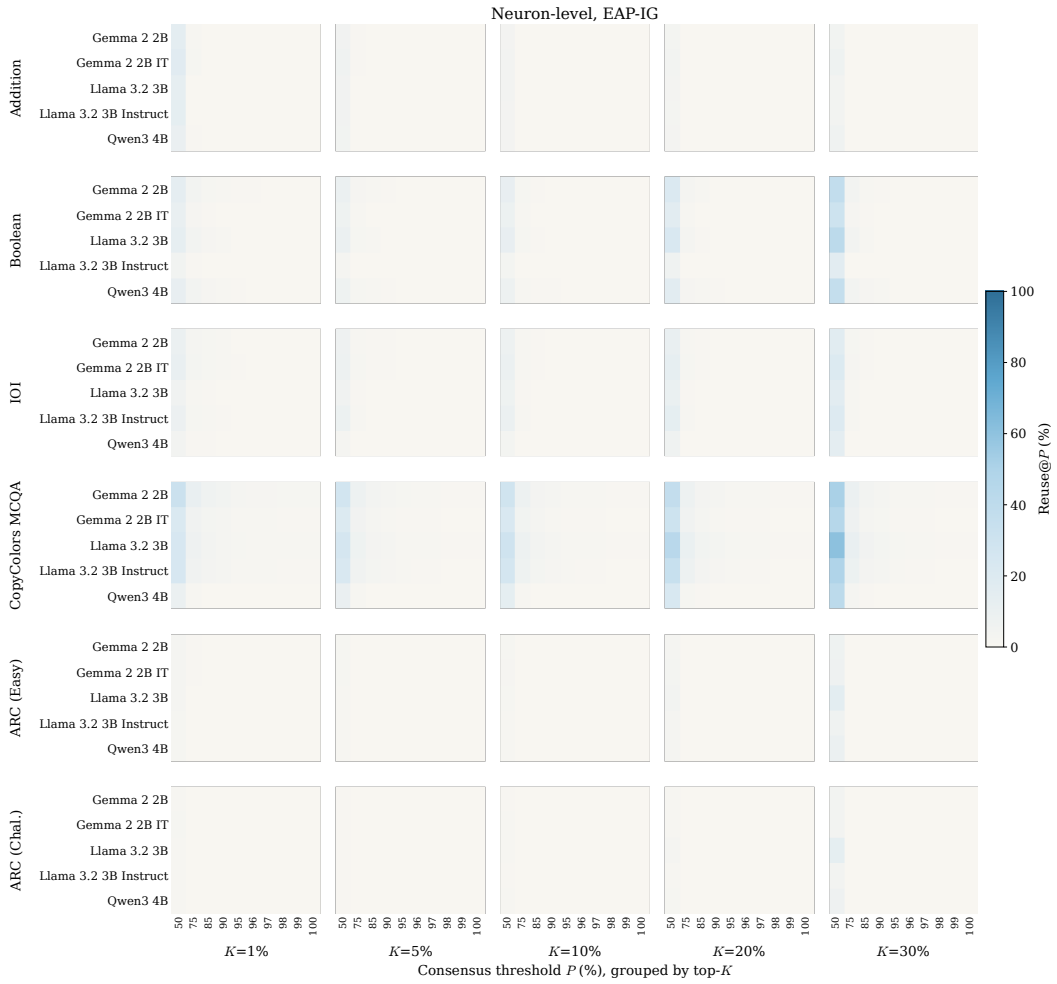


Figure 25: Within-task **Reuse**@P of neuron-level circuits under EAP-IG over the full grid of top-K and consensus threshold P. Columns are grouped by top-K, and rows are grouped by task, with one row per model. A value of 0 includes settings where the task circuit is empty.

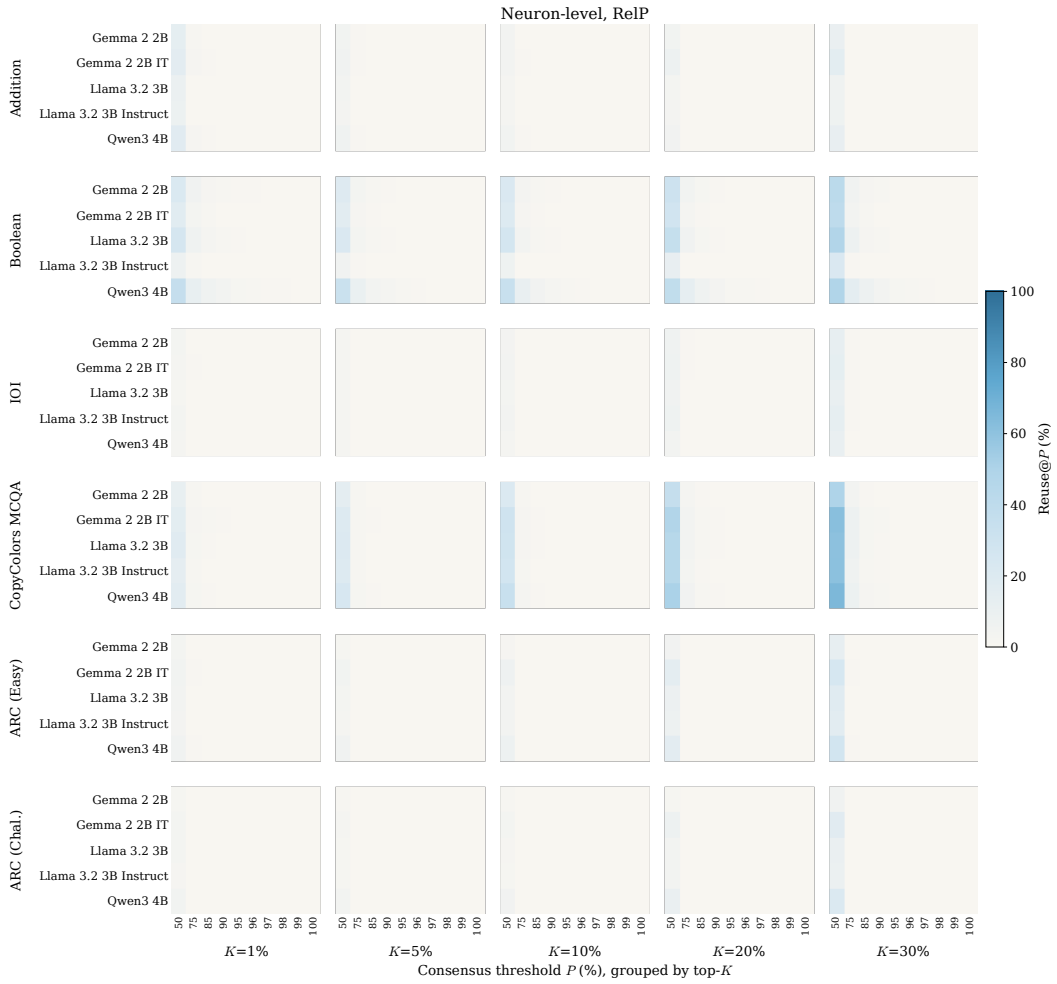


Figure 26: Within-task **Reuse**@P of neuron-level circuits under RelP over the full grid of top-K and consensus threshold P. Columns are grouped by top-K, and rows are grouped by task, with one row per model. A value of 0 includes settings where the task circuit is empty.

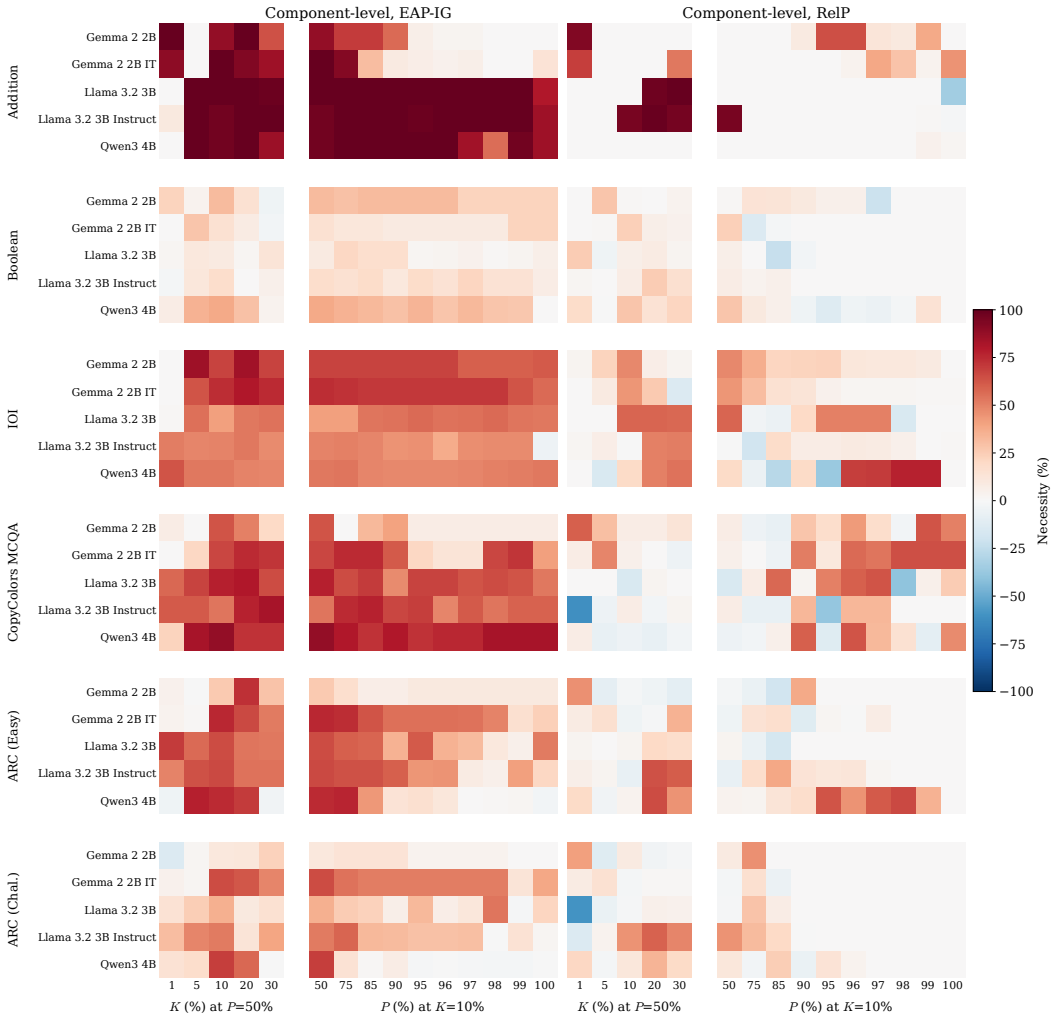


Figure 27: Within-task necessity of component-level circuits under EAP-IG (left) and RelP (right), along top-K at $P = 50\%$ and along P at $K = 10\%$. Rows are grouped by task, with one row per model.

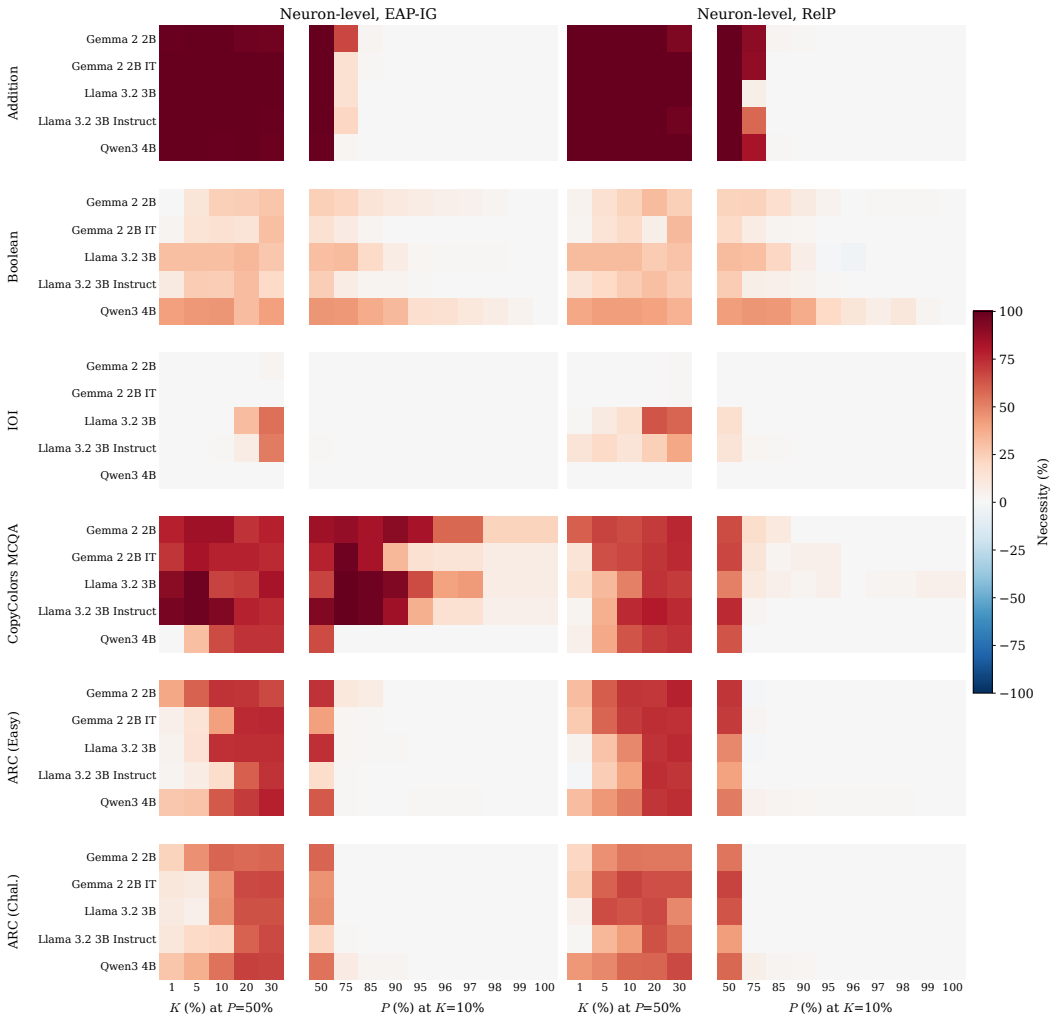


Figure 28: Within-task necessity of neuron-level circuits under EAP-IG (left) and RelP (right), along top-K at $P = 50\%$ and along P at $K = 10\%$. Rows are grouped by task, with one row per model.

L Full Cross-Task Results

In Figure 29 we show the accuracy drop on each task from ablating its own circuit against the mean drop from ablating the other five tasks' circuits, for every model, task and top-K at P=50%, separately for each attribution method and granularity. Component-level circuits lie close to the diagonal from K=5% upwards, except for `IOI` in the `Gemma-2` models at K=5% under EAP-IG and the RelP circuits of the `Gemma-2` models at K=5 and 10%, while at K=1% the circuit of `Addition` is specific in most models. Neuron-level circuits lie above the diagonal at every K, although less so as K grows, with the mean specificity gap falling from 29.4 (EAP-IG) and 26.8 (RelP) points at K=10% to 18.1 and 14.8 points at K=30%.

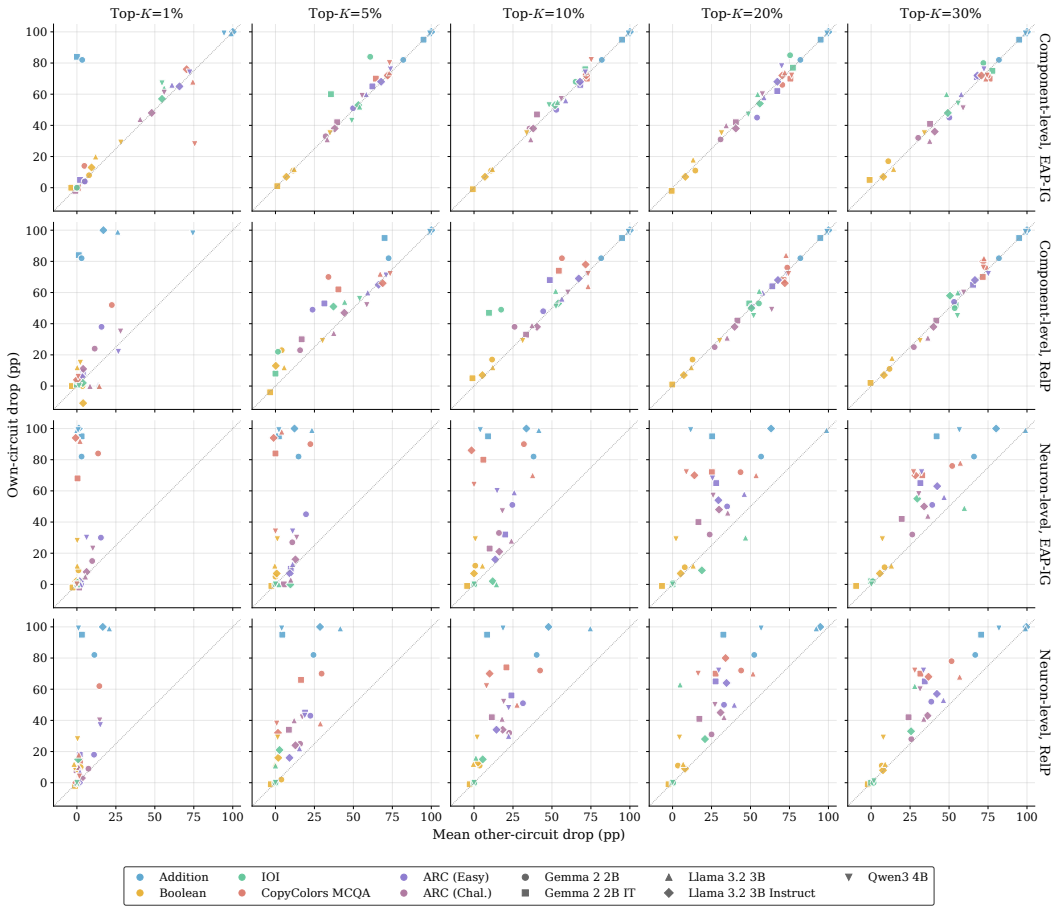


Figure 29: **Own-circuit against mean other-circuit accuracy drop at every top-K, P=50%.** Each point is one model (marker) and task (color). Points above the dotted diagonal are tasks damaged more by their own circuit than by the other tasks' circuits. Rows are the attribution method and granularity, and columns are top-K.

In Figure 30 we show the Jaccard overlap between every pair of task circuits for each model at K=10% and P=50%, the per-model version of Figure 4 (top). We observe that component-level circuits overlap substantially in every model, most of all under EAP-IG in the `Gemma-2` models, where any two tasks' circuits share most of their components (Jaccard overlap of 0.62 to 1.00, with identical circuits for `Boolean Logic` and `IOI`). Overlap is also consistently highest between related tasks, notably the two ARC tasks (0.76 to 0.96 at the component level) and, under RelP, `CopyColors MCQA` and the ARC tasks (0.46 to 0.86), which share the multiple-choice format. Neuron-level circuits, in contrast, barely overlap in any model (at most 0.19), with the two ARC tasks again the only exception (0.35 to 0.56).

Figure 31 shows how the mean overlap over task pairs changes along top-K at P=50% and along P at K=10%, to check that these trends are not specific to one circuit size. The overlap of component-level EAP-IG circuits is stable across both sweeps (0.36 to 0.80 for $K \geq 5\%$), with the exception of `Gemma-2-2B-IT` at P=100% (0.36). RelP overlap, however, shrinks as P grows, from about 0.38 at P=50% to nearly zero at P=100%, as its task circuits shrink (Figure 14). Neuron-level overlap stays low at every K and P (at most 0.12).

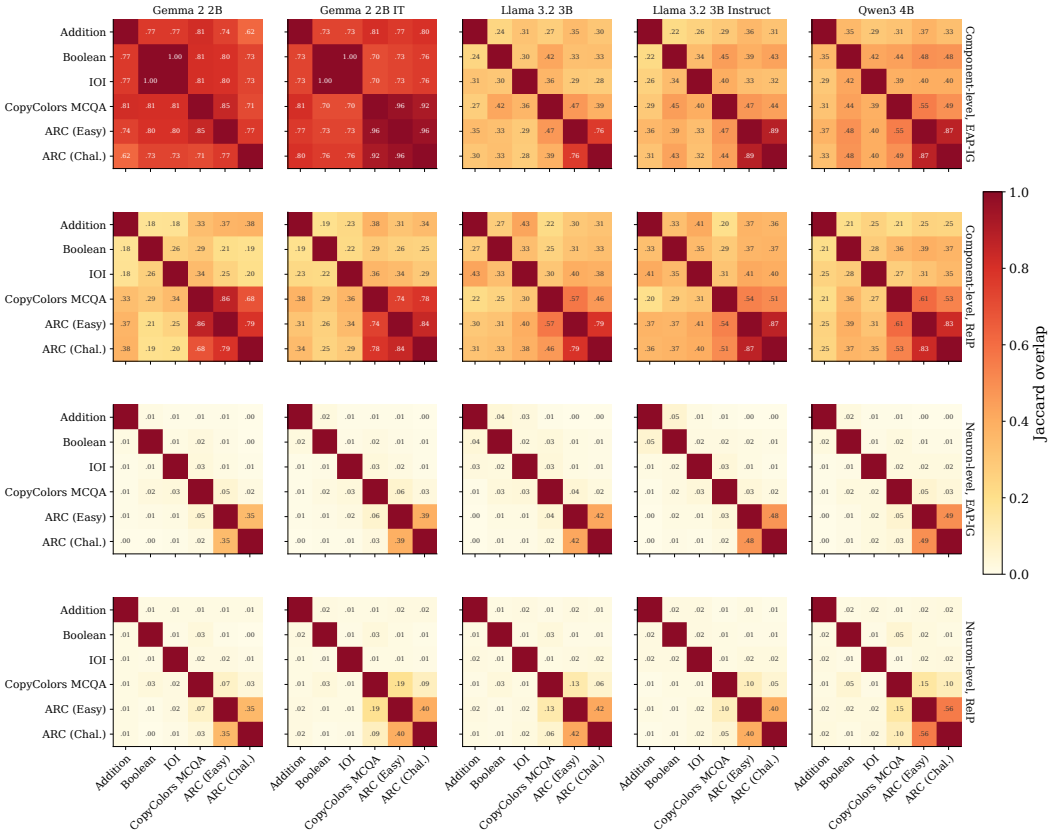


Figure 30: **Jaccard overlap between task circuits for each model at K=10%, P=50%.** Rows are the attribution method and granularity, and columns are models.

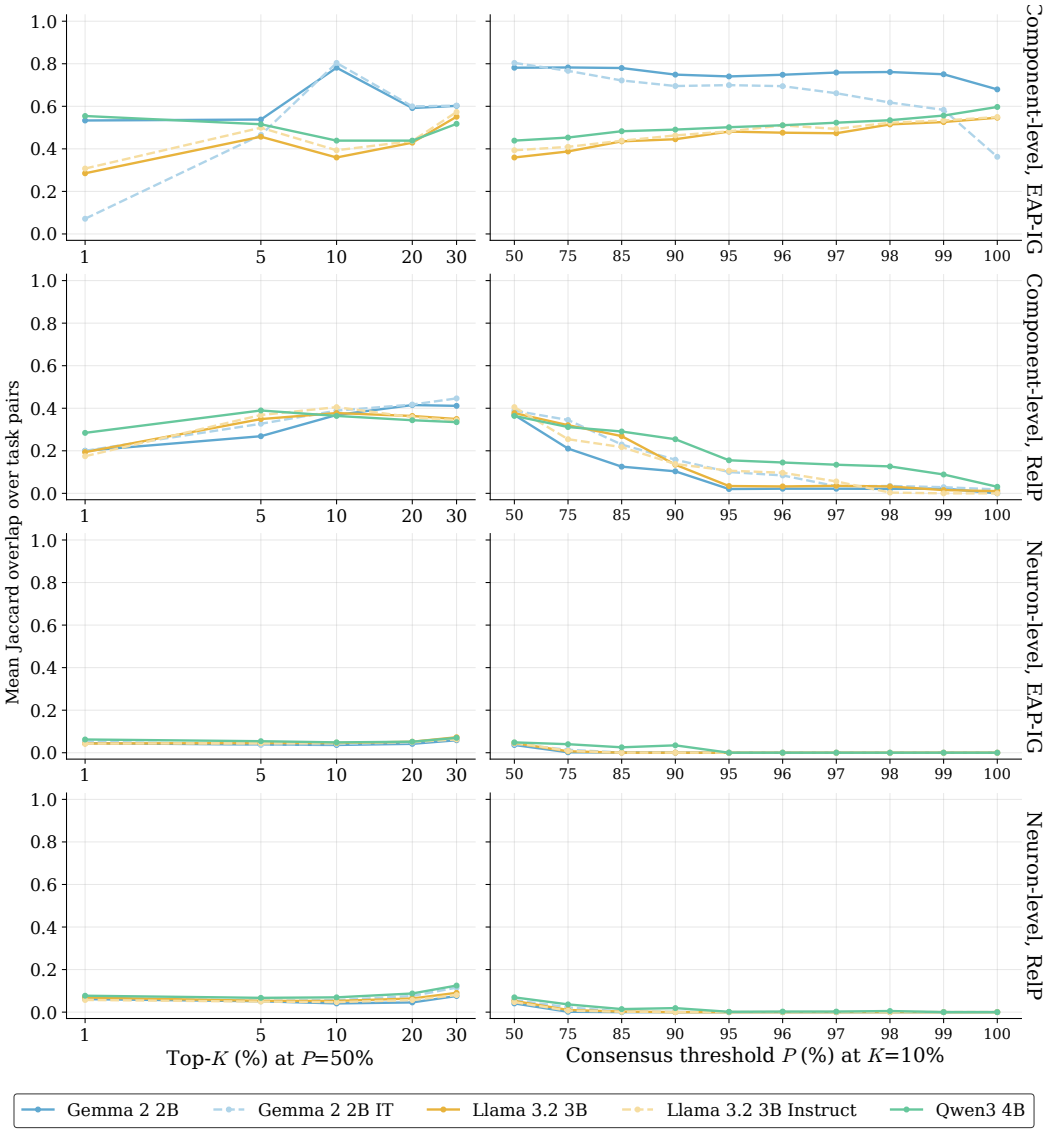


Figure 31: **Mean Jaccard overlap over the 15 task pairs for each model**, along top-K at P=50% (left) and along P at K=10% (right). Instruction-tuned models are dashed. Pairs where both circuits are empty are left out.