SLAYING THE HYDRA: INTERACTION-AWARE CIRCUIT DISCOVERY IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Localizing behavior to individual components of a language model is a central goal of mechanistic interpretability. However, scoring components one at a time misses context-dependent effects: a *primary* component can inhibit the activation of a *backup*, leading to issues with ranking components. *Actual causality* studies the structure of such interactions via *witnesses*: variables that provide contextual information to resolve interaction terms. However, estimation with witnesses typically requires combinatorial enumeration and is infeasible in practice. We introduce the *witness-integrated set effect (WISE)*, a family of causal estimands that build on the witness mechanism while taking expectations over *sets* of causes and witnesses to remain computationally feasible. Building on this approach, we introduce JUNTALEARNER, a gradient-based circuit discovery method that learns to rank components by their causal impact across varying-sized sets of components and witnesses. Alongside faithfulness metrics, we introduce measures of necessity and task specificity, and the *circuit recognition score (CRS)* to summarize each metric across circuit sizes while emphasizing effects achieved by small circuits. Across tasks and models of increasing size, JUNTALEARNER achieves higher mean CRS compared to attribution baselines on all metrics. Since its cost does not grow with the number of candidate components, JUNTALEARNER scales to large models while accounting for set-level interactions and avoiding first-order approximations.

1 INTRODUCTION

Localizing behavior to individual components of a language model is a fundamental goal of mechanistic interpretability (Olah et al., 2020). The standard measures of each component's causal contribution are the *natural indirect effect (NIE)* (Pearl, 2001) and *pure indirect effect (PIE)* (VanderWeele, 2013), often referred to as *noising* and *denoising*, respectively (Heimersheim & Nanda, 2024). Activation patching (Vig et al., 2020; Geiger et al., 2021) computes these quantities exactly, and first-order gradient-based approximations (Hanna et al., 2024; Syed et al., 2024) have made them applicable at scale (Ameisen et al., 2025; Marks et al., 2025).

However, individual NIE scores are systematically distorted by *interactions* between components; genuine causes can be missed and individual effects need not compose into circuit effects (Mueller, 2024). The canonical example is *preemption*: a primary component masks a backup that takes over once the primary is patched (Wang et al., 2023), a form of self-repair that McGrath et al. (2023) call the *Hydra effect*, after the monster that grows new heads for each one cut off (Hesiod, ca. 700 BCE). Interactions can also be *superadditive* or *subadditive*, with joint effects above or below the sum of individual effects (VanderWeele & Knol, 2014). These phenomena reflect a combinatorial structure: the joint effect of any component set includes cross-interactions for every combination of its members, which per-component scores cannot separate (Vaidyanathan et al., 2026).

These failures leave circuit localization caught between Scylla and Charybdis. On one side are per-component scores that ignore interactions. Exact per-component patching needs a forward pass for every component on every prompt, so its cost grows with the model, and it still misses joint effects. First-order approximations are cheaper but discard interactions altogether. On the other side, exhaustive set-level scoring captures interactions but grows exponentially with the number of components, which makes it infeasible for real models. Prior work escapes the search by using prior knowledge to restrict which interactions are considered, either searching for backups to an assumed

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.

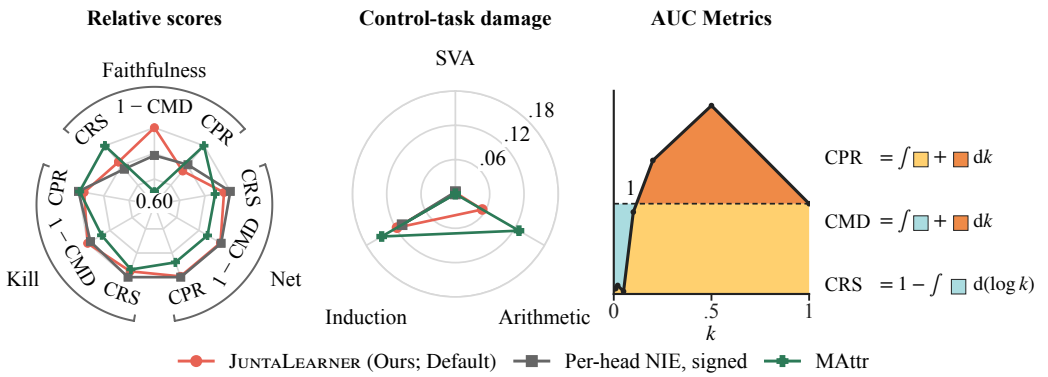


Figure 1: **Interactions and evaluation on GPT-2/IOI. Left:** performance of JUNTALEARNER, compared to the signed per-head NIE and MAttr, across a combination of our metrics. **Center:** as some task (IOI) is successfully "killed" (by removing the estimated 10-head IOI circuit), other tasks may also take collateral damage; this is captured in our "net" metric. **Right:** MAttr faithfulness on GPT-2/IOI incurs significant "overshoot"; CPR rewards this overshoot, while CMD penalizes it. Meanwhile, CRS rewards early discovery by measuring area under the log-curve.

primary circuit (Wang et al., 2023; Gong et al., 2026) or predefining the interaction structure (Chen et al., 2025). What is missing is a method that accounts for interactions at feasible cost.

In this paper, we make the following contributions:

(a) *Witness-Integrated Set Effects (WISE)*, a family of set-level causal estimands parameterized by an effect measure such as NIE, PIE or their soft minimum. WISE scores components by averaging causal effects over *sets* in which they interact, while blocking self-repair from outside the set via *witness pinning*, which holds specific downstream sites at their clean values. Witnesses originate in actual causality (Halpern, 2016); we generalize them to the causal mediation setting.

(b) JUNTALEARNER, a scalable circuit discovery method based on WISE maximization over a learned distribution of candidate sets, through a differentiable Gumbel-top-k relaxation with a dynamically annealed set size. A single run produces a component ranking from which circuits of different sizes can be extracted.

(c) A set of circuit evaluation metrics that expand on the notion of circuit quality beyond faithfulness: *kill*, *net*, and the *circuit recognition score (CRS)*. Kill measures necessity as the task-performance loss when a circuit is patched, while net measures task specificity by penalizing the collateral damage that patching one task's circuit does to other tasks. CRS summarizes each metric across circuit sizes, rewarding early recognition (Truchon & Bayly, 2007) by giving greater weight to high effects achieved by smaller circuits.

(d) On five tasks across GPT-2 small, Qwen3-1.7B and Llama-3.1-8B, JUNTALEARNER outperforms attribution baselines including EAP-IG (Hanna et al., 2024) and MAttr (Arora et al., 2026) on kill and net, and performs at least as well as exact per-head NIE. On the public test split of the MIB Circuit Localization track (Mueller et al., 2025), a single untuned run of JUNTALEARNER beats MAttr's released circuits on kill, net, and the CMD metric for faithfulness that MIB uses to penalize overshooting the original model's performance.

2 WITNESS-INTEGRATED CAUSAL EFFECTS

Counterfactual estimands can give unintuitive answers when causes are redundant, a problem studied at length in actual causality (Halpern, 2016). The standard example is *preemption* (Hall, 2004): Suzy and Billy both throw rocks at a bottle, and Suzy's arrives first and shatters it. Had she not thrown, Billy's rock would have shattered the bottle anyway, so her throw has no counterfactual effect even though it broke the bottle. Language models show the same pattern (Mueller, 2024): GPT-2's IOI circuit has backup name mover heads that were only active once the primary name movers were

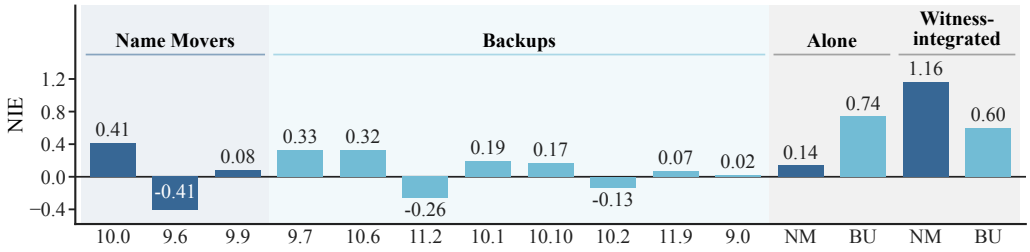


Figure 2: **Witness interactions in the GPT-2 IOI circuit.** The Name Mover group plays a primary role in the GPT-2 IOI circuit, but its backups obscure this role under the standard NIE; witness-integration recovers the intuitive result that the primaries (NM) rank above the backups (BU).

ablated (Wang et al., 2023), and ablating a layer can trigger *self-repair* in the subsequent layers (McGrath et al., 2023).

Indirect Effects Take a factual prompt a, an alternative prompt a', and a set $\mathbf{M}$ of model components, the *mediators* (Steen et al., 2017). The natural indirect effect (NIE) (Pearl, 2001) asks how much of the behavior is lost when $\mathbf{M}$ alone takes its alternative values. The pure indirect effect (PIE) (VanderWeele, 2013) asks how much comes back when $\mathbf{M}$ alone is restored to its factual values:[1]

$$\mathrm{NIE}_{\mathbf{M}}(a, a') = Y(a, \mathbf{M}(a)) - Y(a, \mathbf{M}(a')), \tag{1}$$

$$\mathrm{PIE}_{\mathbf{M}}(a, a') = Y(a', \mathbf{M}(a)) - Y(a', \mathbf{M}(a')). \tag{2}$$

Here $Y(a, \mathbf{M}(a'))$ is the model's output on a with $\mathbf{M}$ patched to the values it takes on a'. The NIE measures necessity and the PIE sufficiency. Interpretability calls them the *noising* and *denoising* directions of activation patching (Heimersheim & Nanda, 2024), and we average both over a task's prompt pairs. Either way the whole set gets a single number, which cannot tell us which members carry the effect or how they depend on one another. That gap matters most when components compensate for each other or cancel out (Wang et al., 2023).

Witnesses *Witnesses* are components downstream of $\mathbf{M}$ that we pin to their values on the starting prompt (a for the NIE, a' for the PIE), so that they cannot compensate for an intervention on $\mathbf{M}$. In the rock throwing example, if we hold fixed that Billy's rock never hits, the bottle now survives whenever Suzy does not throw, and her throw regains its effect. Figure 2 shows the same result in the IOI circuit. Scored alone, the backup name movers outrank the primaries (NIE 0.74 against 0.14); with witnesses integrated, the primaries lead (1.16 against 0.60 for the backups).

With witnesses $\mathbf{W}$, the *witness-informed* NIE and PIE are:

$$\mathrm{NIE}_{\mathbf{M};\mathbf{W}}(a, a') \triangleq Y(a, \mathbf{M}(a), \mathbf{W}(a)) - Y(a, \mathbf{M}(a'), \mathbf{W}(a)), \tag{3}$$

$$\mathrm{PIE}_{\mathbf{M};\mathbf{W}}(a, a') \triangleq Y(a', \mathbf{M}(a), \mathbf{W}(a')) - Y(a', \mathbf{M}(a'), \mathbf{W}(a')). \tag{4}$$

Each witness is pinned to the value it takes on the prompt the run starts from, which is a for the NIE and a' for the PIE.

Witness Integration Since we rarely know in advance which witnesses to pin, we average over a distribution of witness sets (Urbaniak et al., 2026). Let $\mathbf{V}$ be all the variables of the model and $\mathbf{C} \subseteq \mathbf{V}$ the pool of *candidates* we search over. Each draw splits $\mathbf{V}$ three ways. The mediators $\mathbf{M} \subseteq \mathbf{C}$ get patched. The witnesses $\mathbf{W} \subseteq \mathrm{reach}(\mathbf{M})$ get pinned, where $\mathrm{reach}(\mathbf{M})$ is everything downstream of $\mathbf{M}$ outside $\mathbf{M}$ itself. The remaining free variables $\mathbf{F}$ recompute as usual. For a distribution $p(\mathbf{M}, \mathbf{W} \mid \mathbf{C})$ over mediators and witnesses, we define the *witness-integrated set effect (WISE)* of any set-level score $\mathcal{S}$ as

$$\mathrm{WISE}_{p,\mathcal{S}}(a, a') \triangleq \mathbb{E}_{p(\mathbf{M},\mathbf{W} \mid \mathbf{C})}\left[\mathcal{S}_{\mathbf{M};\mathbf{W}}(a, a')\right]. \tag{5}$$

[1]Taking a as the treated level and a' as the control, our PIE is the pure indirect effect of VanderWeele (2013), which equals the NIE of Pearl (2001), and our NIE is the total indirect effect of VanderWeele (2013), which holds the treatment at its active level. We keep the names NIE and PIE to distinguish the two patching directions.

Setting $\mathcal{S} = \mathrm{NIE}$ gives the *witness-integrated NIE*, $\mathrm{WINIE}_p \triangleq \mathrm{WISE}_{p,\mathrm{NIE}}$, the objective that our JUNTALEARNER usually maximizes (Section 3), and $\mathcal{S} = \mathrm{PIE}$ gives $\mathrm{WIPIE}_p \triangleq \mathrm{WISE}_{p,\mathrm{PIE}}$. [2]

Sets and witnesses expose *interaction effects*, the part of a joint effect that per-head evaluation typically misses. When a primary component and its backup fall in the same mediator set, patching them together exposes their joint effect; when the backup is left outside the mediator set, pinning it as a witness does the same. Set sampling and witness pinning are partly overlapping ways of exposing interactions, and how much witnesses add depends on the model; in our experiments it helps most on Qwen3. See Section 4 and Figures 17 and 19 in Appendix D for details.

Enumerating mediator and witness sets becomes impractical as the candidate pool grows. Monte Carlo sampling over set sizes and witnesses can approximate WISE; details are inAppendix A. However, every candidate needs many sampled sets, each of which gets a forward pass, so it scales poorly to large models. In the next section, we describe a gradient-based learner that uses WINIE maximization to identify circuits more efficiently.

3 CIRCUIT DISCOVERY WITH JUNTALEARNER

Since scoring every set of candidates is out of reach, JUNTALEARNER learns which sets are worth scoring. Each candidate has a trainable inclusion logit. At each step, the method draws sets from these logits, scores each set on paired prompts, and updates the logits to favor sets with higher scores. The set size starts large and shrinks as training proceeds. At the end, a readout of the run ranks the candidates, and the top ℓ form the circuit of size ℓ. Appendix B gives the full procedure.

Objective Let θ_i be the inclusion logit of candidate $C_i \in \mathbf{C}$. At set size k, the logits and the witness-sampling rule define a distribution $p_{\theta,k}(\mathbf{M}, \mathbf{W} \mid \mathbf{C})$ over mediator–witness pairs with $|\mathbf{M}| = k$. Training maximizes the witness-integrated score of Section 2 under that distribution:

$$\max_{\theta} \ \mathrm{WISE}_{p_{\theta,k},\mathcal{S}}(a, a') = \max_{\theta} \ \mathbb{E}_{p_{\theta,k}}\big[\mathcal{S}_{\mathbf{M};\mathbf{W}}(a, a')\big]. \tag{6}$$

With $\mathcal{S} = \mathrm{NIE}$ this is WINIE, taken over sets of size k. The choice of $\mathcal{S}$ decides what the ranking looks for: NIE trains toward necessity, and PIE toward sufficiency. We may also obtain a hybrid score by taking the soft minimum of NIE and PIE, which we call the *bottleneck score*:[3]

$$\hat{\mathcal{S}}_{\mathbf{M};\mathbf{W}}(a, a') \triangleq -\tau \log\Big(e^{-\mathrm{NIE}_{\mathbf{M};\mathbf{W}}(a,a')/\tau} + e^{-\mathrm{PIE}_{\mathbf{M};\mathbf{W}}(a,a')/\tau}\Big), \qquad \tau > 0, \tag{7}$$

A candidate's gradient tracks how the expected score changes as that candidate becomes more likely to be drawn.

Sampling and updates Each step takes P prompt pairs and draws B mediator sets from the logits. A set is drawn by adding independent Gumbel noise to every logit and keeping the k largest:

$$g_i \overset{\text{iid}}{\sim} \mathrm{Gumbel}(0, 1), \qquad \mathbf{M} = \{C_i : i \in \mathrm{top}_k(\theta + g)\}. \tag{8}$$

For each mediator set, we sample witnesses uniformly from its eligible downstream candidates, with the witness count determined by a chosen fraction m of that eligible pool. With $m = 0$ no witnesses are pinned, and the score is the plain set-level NIE or PIE. All B sets are scored on the same prompt pairs. The forward pass uses the hard selection, so every component is either patched, pinned or left free to recompute. The backward pass uses a straight-through estimator (Jang et al., 2017), which takes its gradient from the continuous Gumbel-top-k relaxation of Xie & Ermon (2019). The Adam optimizer with gradient clipping then updates the logits.

Shrinking the set size Training follows a descending ladder of set sizes $k_1 > \cdots > k_R$, carrying the logits and the optimizer state from one rung to the next. Large sets allow training to find effects that appear only when several components are patched together, and the smaller sets that follow

[2] Per-component scores can be read off WISE by contrasting sampled sets with and without a component; see Appendix A for details.

[3] As $\tau \to 0$ it approaches $\min(\mathrm{NIE}, \mathrm{PIE})$; for large τ it behaves like their mean shifted by $-\tau \log 2$ and tends to $-\infty$, so τ should be small relative to the scale of the effects.

Algorithm 1 JUNTALEARNER

Require: Candidate pool $\mathbf{C}$; score $\mathcal{S}$; ladder $k_1, \ldots, k_R$; step budget T; B sets and P prompt pairs per step; witness fraction $m \in [0, 1]$
Ensure: $\mathrm{score}(i)$ for every candidate $C_i \in \mathbf{C}$

$\theta \leftarrow \mathbf{0}$ ▷ one inclusion logit per candidate
for $r = 1, \ldots, R$ **do**
  while budget remains and ($r = R$ or rung r has not ended) **do**
    Take P prompt pairs without replacement
    for $b = 1, \ldots, B$ **do**
      $g \sim \mathrm{Gumbel}(0, 1)^{|\mathbf{C}|}$
      $\mathbf{M}_b \leftarrow \{C_i : i \in \mathrm{top}_{k_r}(\theta + g)\}$ ▷ straight-through gradients
      $\mathcal{E}_b \leftarrow$ components downstream of $\mathbf{M}_b$
      $\mathbf{W}_b \sim \mathrm{Unif}\big(\{W \subseteq \mathcal{E}_b : |W| = \lfloor m|\mathcal{E}_b| \rceil\}\big)$
    end for
    $\gamma \leftarrow -\widetilde{\nabla}_\theta \frac{1}{B} \sum_{b=1}^{B} \widehat{\mathcal{S}}_{\mathbf{M}_b;\mathbf{W}_b}$; record γ
    $\theta \leftarrow \mathrm{Adam}(\theta, \gamma)$ ▷ with gradient clipping
  end while
end for
$\mathrm{score}(i) \leftarrow \sum_t \mathrm{clip}(\hat{\gamma}_{t,i})$, over the recorded γ_t normalized per step ▷ Equation (9)

help separate the members of a good set. A rung ends early once its top candidates have stopped changing from step to step, without flipping back and forth, and the logit gap at the k_r-th place is still widening; see Definition B.1 for details. Otherwise it ends after $\lfloor T/R \rfloor$ steps of the total budget T. The last rung runs until the budget is spent, so it also gets the steps that earlier rungs saved. In our experiments, training over decreasing set sizes improves all evaluation scores on every model compared with training only on single-component sets throughout (Section 4).

Ranking Since Adam adjusts the update scale separately for each inclusion logit, final logits need not preserve the relative magnitudes of candidates' effects and hence would make a poor ranking. We therefore rank candidates using gradients recorded during training, normalized per step and clipped per candidate (Equation (9)). The normalization and clipping are engineering choices that are not dictated by the estimands in Section 2; we describe them and how we set them in Appendix B. Writing $C_{(1)}, \ldots, C_{(n)}$ for the ranked candidates, the circuit of size ℓ is $\{\mathcal{C}_{(1)}, \ldots, \mathcal{C}_{(\ell)}\}$, and one training run yields the whole nested family of circuits. Algorithm 1 summarizes the procedure, with $\widehat{\mathcal{S}}$ for the score averaged over the step's prompt pairs and $\widetilde{\nabla}_\theta$ for the straight-through gradient.

4 EXPERIMENTS

We test how well JUNTALEARNER can localize task behavior to compact circuits across models and tasks. We evaluate on GPT-2 small, Qwen3-1.7B-Base, and Llama-3.1-8B, using their attention heads as candidate causes and witnesses. The outcome Y is the correct-answer logit minus the incorrect-answer logit at the final position. For each model, we study performance on indirect object identification (IOI; Wang et al., 2023), subject–verb agreement (Finlayson et al., 2021), two-digit addition (Stolfo et al., 2023), induction (Elhage et al., 2021; Olsson et al., 2022), and entity tracking (Prakash et al., 2024). We provide experimental settings and examples of each task in Appendix C.

We compare JUNTALEARNER against rankings obtained by single-component activation patching (per-head NIE), EAP (Syed et al., 2024), and EAP-IG (Hanna et al., 2024), evaluating rankings by both signed scores and absolute score magnitudes. We also compare against Matryoshka Attribution (MAttr; Arora et al., 2026) and random rankings. The MIB Circuit Localization track (Mueller et al., 2025) gets a separate comparison in Appendix E.

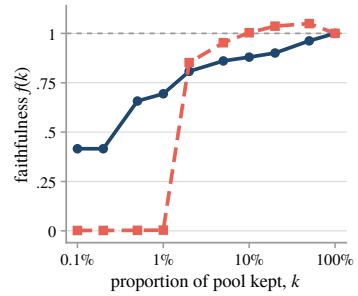
4.1 CIRCUIT EVALUATION

Recovering the model's task performance is the most well-known dimension of circuit quality. *Faithfulness* measures this by ablating components outside the circuit and testing how much performance

remains (Wang et al., 2023), and faithfulness across circuit sizes is often summarized using aggregate metrics such as CPR and CMD (Mueller et al., 2025). While faithfulness establishes how much performance the circuit can support, it does not tell us how much the intact model depends on it: components elsewhere may compensate when the circuit is removed. Moreover, a large removal effect may reflect damage to general-purpose components that support many tasks. We also ask how few components are needed to achieve these effects, to reflect the goal of localizing behavior to small circuits. We therefore supplement standard circuit metrics with measures of removal effects and task specificity, and aggregates across circuit sizes that give greater weight to small circuits.

Faithfulness and Necessity Let the circuit $\mathcal{C}_\ell$ contain a ranking's top ℓ candidates, and write $Y_{\mathcal{C}_\ell} = Y(a, \mathcal{C}_\ell(a'))$ for the outcome when components in $\mathcal{C}_\ell$ are patched to their alternative-prompt values. Faithfulness measures the performance retained when the circuit's complement is patched, while *kill* measures necessity, the effect of patching the circuit itself. In both cases, the remaining components recompute normally, with no witnesses pinned. Then:

$$\text{Faithfulness}(\ell) = \frac{\mathbb{E}[Y_{\mathbf{C}\setminus\mathcal{C}_\ell}] - \mathbb{E}[Y_{\mathbf{C}}]}{\mathbb{E}[Y_{\varnothing}] - \mathbb{E}[Y_{\mathbf{C}}]}, \qquad \text{Kill}(\ell) = \frac{\mathbb{E}[Y_{\varnothing}] - \mathbb{E}[Y_{\mathcal{C}_\ell}]}{\mathbb{E}[Y_{\varnothing}] - \mathbb{E}[Y_{\mathbf{C}}]}.$$



	CPR	1−CMD	CRS
mc, held	0.94	0.94	**0.76**
trained, free	**1.01**	**0.95**	0.59

Figure 3: Two faithfulness curves on Qwen3-1.7B-Base, SVA, at MIB's ten log-spaced proportions. CPR and $1 - \text{CMD}$ prefer the late, overshooting curve; CRS prefers the early one.

All expectations are over held-out prompt pairs, and both metrics are normalized by the effect of patching the full candidate pool. The numerator of kill is the circuit's NIE. Faithfulness reaches 1 when the circuit recovers the intact model's mean outcome; kill reaches 1 when removing the circuit has the same effect as patching the full pool.

Task Specificity To measure task specificity, we apply the same circuit $\mathcal{C}_\ell$ to the set $\mathcal{T}$ of the model's other four tasks. Let $\text{Kill}_j(\ell)$ denote the removal effect on task j using its own prompt pairs and normalization. We define *net* as target-task removal minus mean collateral damage:

$$\text{Net}(\ell) = \min\{1, \text{Kill}(\ell)\} - \frac{1}{|\mathcal{T}|}\sum_{j\in\mathcal{T}} \max\{0, \text{Kill}_j(\ell)\}.$$

We cap the target effect at one so that removing the circuit earns no extra credit for causing more damage than patching all candidate components. We floor each control effect at zero so that improving one task cannot offset damage to another. Higher net values indicate greater target-task disruption relative to damage on the other tasks.

Aggregation Across Circuit Sizes A ranking localizes behavior more compactly when small circuits already achieve the desired effect. Faithfulness is usually summarized using the integrated circuit performance ratio (CPR), the area under the faithfulness curve, and the integrated circuit-model distance (CMD), the area between that curve and one (Mueller et al., 2025). CPR rewards greater recovered performance, including performance that overshoots the intact model's, whereas CMD penalizes deviations in either direction (lower CMD is better; in figures we plot $1 - \text{CMD}$, so that higher is better on every summary). Both integrate over circuit size on a linear scale, so performance at circuit sizes of 1–2% of the candidate pool receives the same weight as performance at sizes of 50–51%, where nearly every ranking has already recovered the behavior.

To prioritize small circuits, we adopt the principle of rewarding early success used in discounted cumulative gain for information retrieval (Järvelin & Kekäläinen, 2002) and early-recognition metrics for virtual screening (Truchon & Bayly, 2007), and average performance over circuit sizes on a logarithmic scale. For each evaluation curve h, we define its *circuit recognition score* (CRS) over evaluated sizes $\ell_1 < \cdots < \ell_J$, with $\ell_1 > 0$, as

$$\text{CRS}(h) = \frac{\sum_{j=1}^{J} w_j \, \min\{1, h(\ell_j)\}}{\sum_{j=1}^{J} w_j}.$$

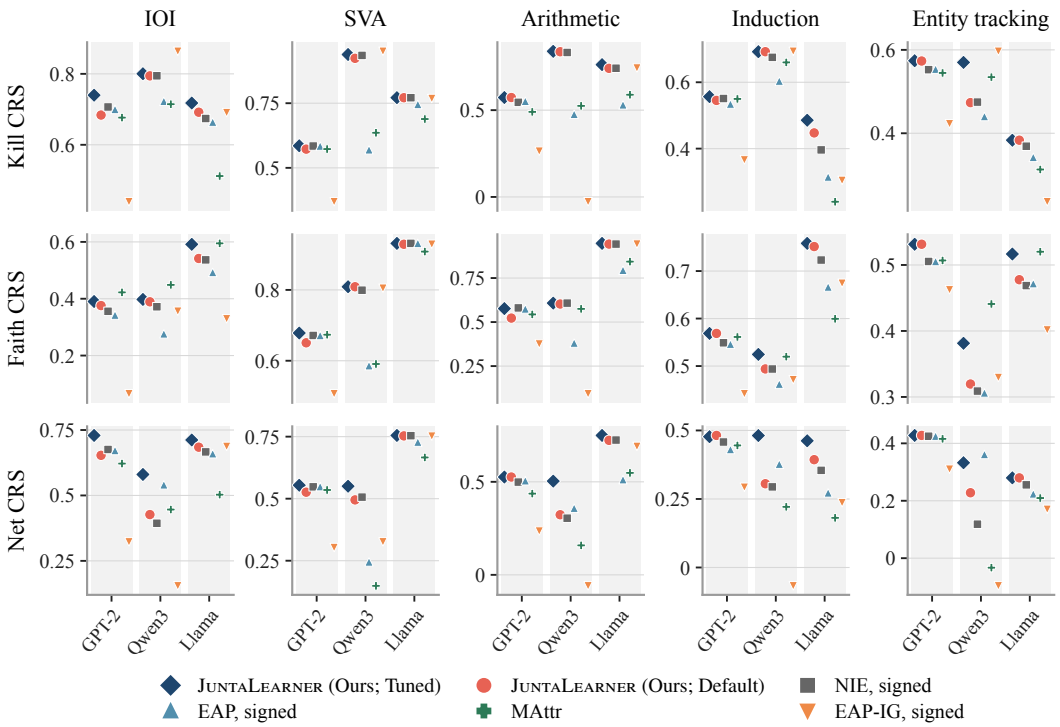


Figure 4: Every (model, task) cell, scored by CRS. *Per-model default* is one configuration per model, chosen by median per-cell score; *Tuned* is, for each cell and each goal, the JUNTALEARNER configuration with the highest CRS on that goal (mean over seeds)

The weights w_j are the local spacings of $\log \ell_j$, computed by central differences at interior points and one-sided differences at the endpoints. Faithfulness is evaluated at MIB's ten log-spaced proportions of the candidate pool, and kill and net at the top $\ell = 1, \ldots, 25$ heads (26 on some cells). Capping faithfulness and kill at one gives no additional credit or penalty for exceeding their reference outcomes; net is already bounded above by one by construction. Applying CRS to the three curves gives S_{faith}, S_{kill}, and S_{net}, respectively.

Figure 3 shows how the choice of summary can reverse a verdict. One curve recovers the behavior from the smallest circuits onward; the other stays at zero until about 1% of the pool and then overshoots. CPR and $1 - \text{CMD}$ rank the late curve higher (1.01 and 0.95 against 0.94 and 0.94), and CRS ranks the early one higher (0.76 against 0.59). We report CRS in the main text; Appendix E reports all three summaries side by side.

4.2 RESULTS

We evaluate two configuration-selection strategies across the fifteen model–task combinations. *Default* uses one configuration per model, selected by its median score across the five tasks. *Tuned* uses task-specific configurations selected from the sweep (Appendix C). We compare the resulting rankings on faithfulness, kill, and net, summarized by CRS.

JuntaLearner outperforms attribution baselines on average. The tuned configurations achieve the highest mean faithfulness, kill, and net scores, leading five of six columns in Table 1. With one fixed configuration per model, JUNTALEARNER scores comparably to signed exact per-head NIE and higher than EAP, EAP-IG, and MAttr. Worst-cell net scores separate the methods further: MAttr, both EAP-IG variants, and magnitude-ranked EAP each have at least one negative-net cell. Figure 4 shows per-cell scores: on GPT-2 small, our default, signed per-head NIE, and signed EAP are within a few hundredths on nearly every panel; on Llama-3.1-8B, JUNTALEARNER default exceeds all attribution baselines on each metric when pooled across tasks. Figure 5 reports margins over signed per-head NIE with 95% intervals across seeds.

Method	Kill (↑) Mean	Kill (↑) Worst	Faithfulness (↑) Mean	Faithfulness (↑) Worst	Net (↑) Mean	Net (↑) Worst
JUNTALEARNER (Ours; Tuned)	**0.67**	**0.38**	**0.61**	0.38	**0.54**	**0.28**
JUNTALEARNER (Ours; Default)	0.65	**0.38**	0.59	0.32	0.48	0.21
Per-head NIE, signed	0.64	0.37	0.59	0.31	0.47	0.12
EAP, signed	0.56	0.31	0.53	0.28	0.46	0.22
MAttr	0.55	0.24	0.58	**0.42**	0.37	-0.03
Per-head NIE	0.56	0.29	0.50	0.22	0.39	0.08
EAP-IG, signed	0.51	-0.03	0.48	0.07	0.28	-0.10
EAP	0.39	-0.07	0.43	0.15	0.31	-0.12
EAP-IG	0.44	0.13	0.44	0.24	0.19	-0.11

Table 1: CRS across the fifteen model-task cells. Worst is the lowest score across all cells; higher is better in every column. Best results are bold and second-best results are underlined, ranked before rounding. Baselines marked "signed" order heads by signed score instead of magnitude. Figure 4 shows the per-cell variation in scores.

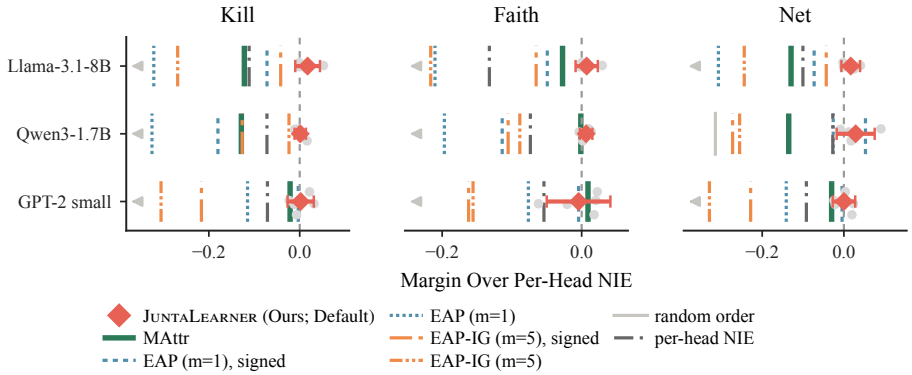


Figure 5: Each model's default against signed per-head NIE, pooled over its five tasks, with a 95% interval across tasks. Vertical rules mark the other baselines' margins on the same axis. Arrows denote that random order is lies significantly off-axes, and would distort the scale to be included.

The baselines' strengths are metric-specific. On Qwen3-1.7B, signed EAP-IG matches the strongest methods on kill for four of five tasks, yet its net score is negative on three—its top-ranked heads recur across tasks (mean IoU 0.28 vs. 0.12 for exact per-head NIE). MAttr shows the opposite profile: highest worst-cell faithfulness (0.42), beating our default on faithfulness in eight of fifteen cells, while our default exceeds it on kill in twelve and on net in thirteen. These differences reflect training objectives: MAttr patches the complement of the circuit, which is what faithfulness measures; JUNTALEARNER patches the circuit itself, which is what kill measures.

Training is faster than exact scoring on the largest model. On Llama-3.1-8B, JUNTALEARNER trains in 14–16 minutes on the slowest task, versus 97 minutes for exact per-head NIE—a 4–7× speedup that holds across all timed runs (Figure 6). On the smaller models, training is 2–8× slower than exact patching, which takes seconds to minutes. The difference is structural: exact scoring costs $\Theta(|M| \cdot N)$ forward passes, scaling with the candidate pool; training costs $\Theta(T \cdot B \cdot P)$, independent of pool size.

JuntaLearner tuning favors descending set sizes and necessity objectives, with model-specific gains from witness pinning. Each run is shaped by five knobs: whether to descend a ladder of set sizes or train on single candidates throughout, how wide to start the ladder (as a share of the candidate pool), which objective to score (NIE, NIE + PIE, or the bottleneck score of Equation (7)), how many downstream witnesses to pin (none, half, or all), and whether to add a final measurement stage that scores leading candidates individually. We fit a Gaussian process to each model's training runs and select settings by expected improvement, double-checked by CatBoost (Figure 17). Compared with training on single components throughout, the descending ladder improves predicted faithfulness, kill,

and net on all three models. Starting the ladder with larger sets further improves kill and net, with a small loss in faithfulness, since large early sets favor heads whose removal matters in combination. Scoring sampled sets for necessity alone performs at least as well as combining necessity and sufficiency through either their sum or the bottleneck score. The optional measurement stage has a smaller effect, changing no model–metric comparison by more than 0.4 posterior standard deviations.

The benefit of witness pinning depends on the model and metric. Pinning all eligible witnesses lowers all three scores on GPT-2 and Llama. On Qwen3, it improves net while reducing kill and faithfulness, with normalized changes of $+0.7$, -1.8, and -2.5, respectively. This tradeoff also appears in the task-specific results: configurations with witnesses pinned achieve the highest net scores on all five tasks in Qwen3. This is potentially suggestive of many multi-functional backup heads in Qwen3, though we do not further investigate this claim in this work.

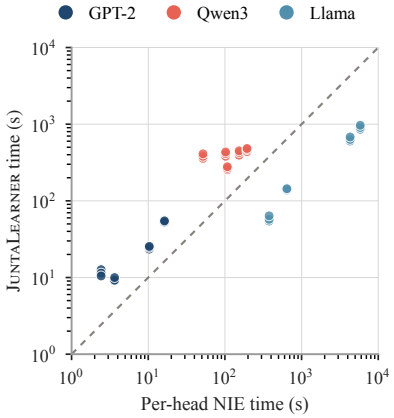


Figure 6: Training time for each model–task–regime run (seconds, log scales). The dashed line marks equal time; points below it favor JuntaLearner.

Evaluation on MIB We extend the comparison to the MIB Circuit Localization track (Mueller et al., 2025), covering twelve model–task combinations across four models and six tasks. We compare JuntaLearner with exact per-node NIE and MAttr's released circuits on the public test split, with one shot untuned runs. Faithfulness uses MIB's native evaluator over circuits containing attention heads, MLPs, and the input embedding; we additionally evaluate kill and net over the attention heads in each ranking. We selected necessity as the score for training sets using GPT-2 IOI validation data; further protocol details and sensitivity analyses appear in Appendix E.

Across the twelve cells, JuntaLearner achieves better aggregate kill and net scores than MAttr under all three summaries, and exceeds exact per-node NIE on CRS for faithfulness, kill, and net. MAttr's mean faithfulness curve rises earlier, giving it a higher CRS, and its higher CPR comes mostly from overshoot (Figure 23). JuntaLearner's circuits do far better on CMD since it penalizes overshoot: summed over the twelve cells, their $1 - \mathrm{CMD}$ is 9.1 against MAttr's 1.0 (Figure 22).

5 Discussion

In the presence of interactions, recovering a behavior and selectively disabling it need not lead to the same circuit. A set of components can support the behavior while backups elsewhere preserve it after removal; removing a different set can disrupt the behavior through damage to machinery shared across tasks. WISE and JuntaLearner account for how components contribute in the context of other sets of components. The clearest sign that component sets matter comes from training itself: the descending ladder improves scores over singleton training on all metrics. Additionally, JuntaLearner's higher net scores indicate circuits that are more localized to the specific task being studied, which helps with controlling its behavior in isolation.

However, the choice of evaluation metric can reverse a verdict on circuit quality. Strong removal effects alone do not establish task specificity; several baselines achieve competitive kill scores yet negative net scores. MAttr matches or exceeds the JuntaLearner default on faithfulness in about half of cases, particularly with CPR, but much of its lead comes from overshoot and CMD penalizes this. Our default is also ahead on kill and net in the majority of cases. These findings motivate reporting necessity and task specificity alongside faithfulness, with multiple summaries of each curve.

Several limitations remain. Our main experiments focus only on attention heads, and on MIB we ran a single untuned configuration. Training is slower than exact per-head patching on the smaller GPT-2 and Qwen3 models, and the cost advantage appears only once the pool is large. The benefit of witness pinning depends on the model and was primarily seen on Qwen3; a principled account of this dependence would clarify when and why witnesses will help identify context-dependent mechanisms.

AI USE STATEMENT

We used generative AI assistants (Anthropic's Claude Sonnet 5 and Opus 5, through the Claude Code CLI, as well as GPT-6 Astra through the Codex CLI) in the engineering and experimental work behind this paper. The assistants ran the experimental sweeps: they provisioned and drove the GPU jobs, monitored them to completion and collected the results. They reformatted the published task datasets (entity tracking, IOI, SVA, arithmetic, induction). They also took part in experimental design in one respect. Human designed Gaussian-process and CatBoost surrogates, fit to the runs already finished, chose which combination of model, task and objective to run next, and the assistants ran that loop: they fit the surrogates and carried out the runs the recommendations pointed to. The authors set the objective and approved every run.

We did not use generative AI to implement the methods this paper is about. The estimators, the samplers, the junta learner and the witness machinery were written by the authors; the assistants debugged and refactored that code and wrote the experiment drivers around it. Interpretation is likewise ours. The authors decided what the numbers mean and every claim we draw from them; the assistants turned experimental data into figures and internal summaries, to specifications we set. The conceptual framework is ours too. No assistant proposed or refined the generalized causal effects estimand, the kill and faithfulness metrics of Section 4, or the hypotheses our experiments test. Nor did we generate synthetic data with an assistant; every task dataset is built by template from published sources. The remaining categories requiring disclosure do not apply to this work: the paper states no theorems and contains no proofs, no translation was involved, and the work has no qualitative or thematic data to analyse.

Among the categories where disclosure is recommended, the assistants edited and debugged code throughout the repository, wrote the figure- and table-generating scripts in the released code, searched for related literature, formatted bibliography entries, and drafted early versions of some paragraphs of this paper, in each case from notes in which an author had already described the ideas to be contained in it. Every such paragraph was rewritten or edited by an author before it entered the text. The assistants also proofread the finished prose, reporting typos and claims that disagreed with our data and experimental results.

We have reviewed all AI-assisted work. Every number and every figure in the paper is regenerated by a script in the codebase from saved run artifacts. We take responsibility for the final content of this work, including artifacts produced with the aid of generative AI.

REFERENCES

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.

Aryaman Arora, Kirill Acharya, Nathan Hu, Yanzhe Zhang, Noah Goodman, Dan Jurafsky, and Christopher Potts. Matryoshka attribution: Learning to attribute language model outputs to representations and weights, 2026. URL https://arxiv.org/abs/2609.25518.

Hang Chen, Jiaying Zhu, Xinyu Yang, and Wenya Wang. Rethinking circuit completeness in language models: AND, OR, and ADDER gates. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=1WkolvkTVi.

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. https://transformer-circuits.pub/2021/framework/index.html.

Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart Shieber, Tal Linzen, and Yonatan Belinkov. Causal analysis of syntactic agreement mechanisms in neural language models. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1828–1843, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.144. URL https://aclanthology.org/2021.acl-long.144/.

Atticus Geiger, Hanson Lu, Thomas F Icard, and Christopher Potts. Causal abstractions of neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=RmuXDtjDhG.

Zhiren Gong, He Lu, Tiantong Wang, Yichi Zhang, Yixin Wang, Zihao Zeng, Ming Xiao, Chau Yuen, and Wei Yang Bryan Lim. Conditional co-ablation: Recovering self-repair backups in transformer circuits, 2026. URL https://arxiv.org/abs/2607.01940.

Ned Hall. Two concepts of causation. In John Collins, Ned Hall, and L. A. Paul (eds.), *Causation and Counterfactuals*. The MIT Press, 06 2004. ISBN 9780262270663. doi: 10.7551/mitpress/1752.003.0010. URL https://doi.org/10.7551/mitpress/1752.003.0010.

Joseph Y. Halpern. *Actual Causality*. MIT Press, Cambridge, MA, 2016.

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.

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

Hesiod. *Theogony*. ca. 700 BCE.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

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.

Thomas McGrath, Matthew Rahtz, Janos Kramar, Vladimir Mikulik, and Shane Legg. The hydra effect: Emergent self-repair in language model computations, 2023. URL https://arxiv.org/abs/2307.15771.

Aaron Mueller. Missed causes and ambiguous effects: Counterfactuals pose challenges for interpreting neural networks. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024. URL https://openreview.net/forum?id=pJs3ZiKBM5.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Ivan Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fiotto-Kaufman, Tal Haklay, Michael Hanna, et al. MIB: A mechanistic interpretability benchmark, 2025. URL https://arxiv.org/abs/2504.13151.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. 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. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pp. 411–420, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558608001.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *Proceedings of the 2024 International Conference on Learning Representations*, 2024. arXiv:2402.14811.

Johan Steen, Tom Loeys, Beatrijs Moerkerke, and Stijn Vansteelandt. Flexible mediation analysis with multiple mediators. *American Journal of Epidemiology*, 186(2):184–193, 2017. doi: 10.1093/aje/kwx051.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7035–7052, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.435. URL https://aclanthology.org/2023.emnlp-main.435/.

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/.

Jean-François Truchon and Christopher I. Bayly. Evaluating virtual screening methods: Good and bad metrics for the "early recognition" problem. *Journal of Chemical Information and Modeling*, 47(2):488–508, 2007.

Rafal Urbaniak, Sam Witty, Daniel Waxman, Andy Zane, Poorva Garg, Emily Bunnapradist, Sankaran Vaidyanathan, Jack Feser, Drew Lehe, and Eli Bingham. A computationally feasible framework for causal probabilistic explanation, 2026. URL https://arxiv.org/abs/2609.04177.

Sankaran Vaidyanathan, David Arbour, Aaron Mueller, Scott Niekum, and David Jensen. The curse of multiple mediators: Hidden interaction effects in activation patching, 2026. URL https://arxiv.org/abs/2606.27510.

Tyler J. VanderWeele. A Three-way Decomposition of a Total Effect into Direct, Indirect, and Interactive Effects. *Epidemiology*, 24(2):224–232, 2013. ISSN 1044-3983.

Tyler J VanderWeele and Mirjam J Knol. A tutorial on interaction. *Epidemiologic methods*, 3(1): 33–72, 2014.

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.

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.

Sang Michael Xie and Stefano Ermon. Reparameterizable subset sampling via continuous relaxations. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

A SAMPLING ALGORITHM

This paper builds a per-candidate ranking in two ways. The sampled ranking described here is based on a Monte Carlo estimate of the WISE score of Section 2 and some similar scores: it draws candidate sets, scores each one as a set, and reads a candidate's contribution off the difference between the sets that hold it and the sets that do not. The trained ranking of Appendix B, roughly speaking, learns an inclusion distribution and is read off the training trace. Everything in Section 4 uses the trained ranking, as MC is costly for larger models like Llama-3.1-8B. We do use the MC estimands on smaller models to look at the overlaps between the resulting rankings (Appendix B) for the family of estimands introduced in Section 2.

For simplicity we describe the sampler with witnessed NIE as the objective, which can be swapped for any of the others.

Algorithm 2 Sampling. One pass over the sweep, then one score per candidate.

Require: pool $\mathcal{P}$; sizes $\mathcal{K}$ with a draw budget n_k for each; a witness regime
Ensure: $\mathrm{score}(i)$ for every candidate $i \in \mathcal{P}$

$\mathcal{D} \leftarrow \emptyset$ ▷ scored sets
for $k \in \mathcal{K}$ with $k < |\mathcal{P}|$ **do** ▷ $k = |\mathcal{P}|$ splits nothing
  for n_k draws **do**
    $S \sim \mathrm{Unif}\big(\{\, S \subseteq \mathcal{P} : |S| = k \,\}\big)$
    $W \leftarrow$ the regime's witness draw on $\mathrm{reach}(S)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(S, \mathrm{NIE}_{S;W})\}$
  end for
end for
for $i \in \mathcal{P}$ **do**
  $\mathcal{K}_i \leftarrow \{\, k \in \mathcal{K} :$ some draw at size k holds i and some does not $\}$
  $\mathrm{score}(i) \leftarrow \dfrac{1}{|\mathcal{K}_i|} \displaystyle\sum_{k \in \mathcal{K}_i} \Big(\operatorname*{mean}_{\substack{(S,v)\in\mathcal{D} \\ i\in S,\ |S|=k}} v - \operatorname*{mean}_{\substack{(S,v)\in\mathcal{D} \\ i\notin S,\ |S|=k}} v \Big)$
end for

The key element in the above is the matching of sets within a size. A larger set removes more of the margin whoever is in it, so a candidate drawn mostly into large sets would otherwise be credited for the size. The price is that the number is a contrast a candidate with no causal role is not guaranteed a score of zero (only the induced ordering is every used to generate circuits, though).

B LEARNING ALGORITHM

Most of a model's heads do nothing for any given task, and the learner looks for the few that do. It trains one inclusion logit per candidate, saying how likely that candidate is to be switched on, in order to maximize the causal effect. Training then repeats the following steps: draw sets of candidates from those logits (and witnesses, if any), run forward computing the causal effect, and backpropagate to move the logits.

Candidate set size is annealed. Typically, training starts with some set size k_1 broad enough to quickly explore a larger number of candidates at a high level, and ends with k_R, narrow enough that a set's score says something about single candidates. The strategy is that of the twelve-ball balance puzzle, where three weighings find the odd ball only because each one puts groups of decreasing size on the pans. Here k falls for analogous reasons. We call the sequence of sizes a ladder and one size a rung. A run descends the ladder and stays on each rung until that rung is earned or its step budget runs out.

One step of Algorithm 3 goes as follows. It takes P prompt pairs and draws B candidate sets of the rung's size k_r, each set with its own witness subset drawn from what that set reaches. It scores every set with the witness-integrated objective $\mathcal{S}$ on those pairs. It then takes one Adam step on the logits. The draw is a top-k Gumbel sample with a straight-through estimator, which makes a hard set of k_r candidates differentiable in θ. Prompt pairs are taken without replacement from a shuffled pass over

Algorithm 3 Learning. One pass down the cardinality ladder.

Require: pool $\mathcal{P}$; effect score $\mathcal{S}$; ladder $k_1, \ldots, k_R$ in the order the schedule visits them; step budget T; B sets and P prompt pairs per step; witness fraction $m \in [0, 1]$
Ensure: $\mathrm{score}(i)$ for every candidate $i \in \mathcal{P}$

$\theta \leftarrow \mathbf{0}$ ▷ one inclusion logit per candidate
$C \leftarrow \lfloor T/R \rfloor$ ▷ each rung's step ceiling
for $r = 1, \ldots, R$ **do**
  while rung r is neither earned nor at its ceiling **do** ▷ rung R runs until T is spent
    take P prompt pairs without replacement
    for $b = 1, \ldots, B$ **do**
      $g \sim \mathrm{Gumbel}(0, 1)^{|\mathcal{P}|}$
      $S_b \leftarrow \mathrm{top}_{k_r}(\theta + g)$ ▷ straight-through, so $\partial S_b / \partial \theta$ exists
      $W_b \sim \mathrm{Unif}\big(\{\, W \subseteq \mathrm{reach}(S_b) : |W| = \lfloor m\, |\mathrm{reach}(S_b)| \rceil \,\}\big)$
    end for
    record ∇_θ for this step
    $\theta \leftarrow \mathrm{Adam}\Big(\theta,\ -\nabla_\theta \frac{1}{B} \sum_b \mathcal{S}_{S_b; W_b}\Big)$
  end while
end for
$\mathrm{score}(i) \leftarrow$ the readout of Equation (9) over the recorded gradients

the training split, refilled when the pass runs out, so no pair is seen much more often than another. And all B sets in a step score the same pairs.

Advancing a rung. A rung ends one of two ways. Either the logits settle, and the rung is *earned*, or the rung runs out of steps and ends *at its ceiling* (the last rung always runs until the budget is spent). Settling means three things at once: the top of the ranking stops changing from step to step, it is not merely oscillating, and the gap at the cut is still widening.

Definition B.1 (Earned rung)**.** Rung r trains at cardinality k_r over a sequence of steps, each ending with an Adam update to θ. At step t, let $w = \max(k_r, \min(10, |\mathcal{P}|))$ and $\mathrm{supp}_t = \mathrm{top}_w(\theta_t)$, the index set of the top w logits. w is wider than k_r itself: at $k_r = 1$ a one-element set makes overlap binary, reporting only whether the leader changed. The step-to-step overlap is $o_t = \frac{|\mathrm{supp}_t \cap \mathrm{supp}_{t-1}|}{|\mathrm{supp}_t \cup \mathrm{supp}_{t-1}|}$, and the separation margin μ_t is the gap between the k_r-th and $(k_r{+}1)$-th largest entries of θ_t, normalized by the spread between the median of its top fifth and the median of the whole vector.

Let $C = \lfloor T/R \rfloor$ be the rung's step ceiling, $\alpha = \min(1, 3/C)$, and $\mathrm{span} = \max(2, \mathrm{round}(1/\alpha))$, about a third of C. Let $\bar{o}_t$ be the exponential moving average of o_t with decay α. Rung r is *earned* at the first step $t \geq \mathrm{span}$ where the following three conditions are met:[4]

1. **(Convergence)** The moving average of o_t exceeds a threshold τ, i.e., $\bar{o}_t \geq \tau$;
2. **(Stability)** Over a sliding window, o_t has never dropped more than ϵ-far from the threshold τ, defined as $\min(o_{t-\mathrm{span}+1}, \ldots, o_t) \geq \tau - \epsilon$; and
3. **(Separation)** The separation margin μ_t has improved over the beginning of the sliding window, i.e., $\mu_t \geq \mu_{t-\mathrm{span}+1}$.

Readout. First of all, the optimizer in this task has quite a bit to handle. The gradient of a set score has no fixed scale. It depends on the objective, on the model, on how many candidates the set holds and on which other candidates came with them, and it falls several-fold over the course of the ladder as k narrows. Plain gradient ascent at a fixed learning rate therefore behaves differently on every task and every rung. Adam normalization smooths things over and in a sense removes the scale. It keeps a

[4] We use $\tau = 0.9$ and $\epsilon = 0.1$ throughout. Each condition rules out a different way of being fooled. Two runs can reach the same $\bar{o}_t$: one that has settled, and one that flips between two members every other step. Only the flipping one dips below $\tau - \epsilon$ inside the span, and that dip is what the second condition looks for. The third catches a run that has gone quiet instead of resolving, where the top k_r stop changing because the logits have stopped moving at all.

running estimate of each logit's own recent gradient magnitude and divides that logit's update by it, so every logit moves about one learning rate per step whatever its raw gradient happened to be. Only the direction and the consistency of a candidate's gradient survive the division. That is what makes one step budget work across models, tasks and objectives.

It is also what makes the final logit not great as a direct score. Adam normalization divides away the effect size and so candidates can end a run at the same logit for quite different reasons: one had a large effect on a few steps, the other a small effect on every step. Just looking at the logits won't help you resolve this.

Note however that the gradients themselves are still in the trace. We cannot just sum them up: with B sets drawn per step, a single large gradient at a step can be an artifact of the company a candidate happened to be drawn with, and that one step might easily dominate the sum. So the next simple thing one could do is to (1) normalize within step to account for changes in set sizes (and resulting changes in gradient absolute values), and (2) cap the sum do mitigate the impact of extreme bursts. Write $g_t \in \mathbb{R}^{|\mathcal{P}|}$ for the negated gradient recorded at step t (line 11 of Algorithm 3). Normalize each step, then clip each candidate's contribution against its own scale:

$$\hat{g}_t = \frac{g_t}{\max\big(\|g_t\|_2,\ 0.25\,\mathrm{median}_t\|g_t\|_2\big)}, \qquad s_i = \mathrm{median}_t|\hat{g}_{t,i}|,$$
$$\mathrm{score}(i) = \sum_t \mathrm{clip}\big(\hat{g}_{t,i},\ -c\,s_i,\ c\,s_i\big), \tag{9}$$

with $c = 3$, and the median taken over every step, including the steps that did not draw i. The per-step normalization gives every rung one unit-norm vote, because gradient scale falls several-fold over the ladder, and an unnormalized sum would weight the broad early rungs, where nothing is resolved yet, over the narrow late ones where the run settles its top-k. The cap limits one step to c times a candidate's own typical gradient, so a burst adds at most three medians however large it is, while a candidate with real support on most steps keeps accumulating at full strength.

Measuring in readout. Equation (9) is an indirect score. A candidate's gradient is read in the company of whichever other candidates its set happened to hold, and the sum has to be normalized and capped to keep that company from deciding the answer. Where a direct measurement of a candidate's own effect is available, it is the quantity the ranking wants, and no inference is needed. A ladder whose last rung is $k = 1$ makes some of those measurements for free. Every set drawn on that rung holds exactly one candidate, so the set's score is not evidence about that candidate's effect, it is that effect, measured on the step's prompts. So, if the last rung actually happened to be $k = 1$, one trick to test is to use the individual effects from that stage directly to sort the final top candidates. Training still decides who gets measured. Draws on the last rung come from the logits, so a candidate is drawn at $k = 1$ only if the ladder has already carried it near the top. The measurements settle the order among those finalists, assuming the signal-to-noise ratio in this final stage is promising.[5]

Measurement stage in training. A final training rung at $k = 1$, as discussed above, draws from the logits and spends most of its draws on the current leaders, whose correct ordering matters the most for circuit discovery. While that sounds like the right place to spend them, it isn't always the case. The reason is how the draws are spread inside the leading group. It might happen that the candidate in front is drawn again and again, long after its position has stopped being in doubt, so measurement after measurement is spent confirming something already settled. The candidates just behind it might still be drawn once or twice each, and one or two draws cannot order them properly. The allocation is somewhat self-reinforcing: the candidate the logits favour is the candidate the draws keep going to.

The measurement stage conducts some measurements in a more balanced fashion to alleviate those potential undesirable inequalities, and since we already sample them one by one, it corrects by their own individual effects. It focuses on a set of the top $M = \lfloor B\,T'/d \rfloor$ candidates by logit, for B worlds per step, T' steps remaining and d measurements per candidate, with $d = 4$ throughout. Inside that set every candidate is measured the same number of times, the leader and the M-th alike, because the

[5]None of this applies to a ladder whose last rung is larger than one, since no set it draws ever holds a single candidate. That is Llama's case: its default ladder is 142-71-43-21-7, so every set holds at least seven candidates, no direct measurement is ever taken, and the ranking is Equation (9) and nothing else.

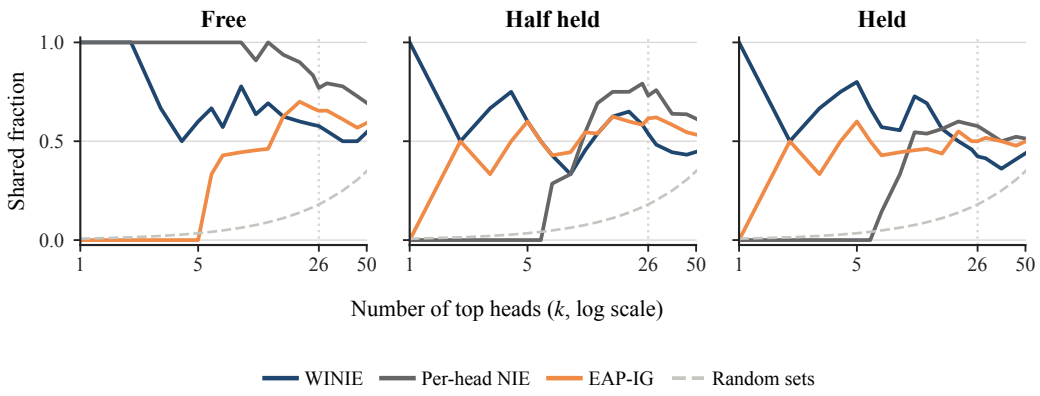


Figure 7: Agreement of three score orderings with the learned JUNTALEARNER ordering for GPT-2/IOI (seed 0). Plots the free, half-held, and held witness regimes, for the first 50 out of 144 attention heads. Each curve gives the fraction of the top k learned heads also in a score ordering's top k. The gray dashed curve is the random-set expectation $k/|\mathcal{P}|$; the dotted vertical line marks $k = 26$. The Qwen held cell was not run.

stage draws them in a balanced cycle instead of drawing from the logits. Candidates outside keep the logits and gradients from the training.

The push starts from the logit each covered candidate had when the stage began. Writing $y_{i,m}$ for candidate i's m-th measurement, the stage keeps a running mean:

$$\theta_i \leftarrow \theta_i + \frac{y_{i,m} - \theta_i}{m}. \tag{10}$$

The first measurement has $m = 1$, so it moves θ_i the whole way onto $y_{i,1}$ and the entry logit drops out.

Algorithm 3 optimizes an inclusion distribution while using a set-based score such as WINIE, and never estimates the WISE score of Section 2. The relation between the resulting rankings and the ones that would have been established using the estimands is interesting. While deeper theoretical connections await exploration, we have partially answered this question empirically (Figure 7). Each panel plots, for one (model, task) cell, the share of a reference ordering's top k that three per-candidate orderings also put in their top k, against the $k/|\mathcal{P}|$ two random sets of that size share. Per-head NIE and WINIE score a candidate alone, with witnesses free and integrated respectively. EAP-IG is the gradient baseline. We find that per-head NIE is most in line with JUNTALEARNING when witnesses are free, and that WINIE comes further into agreement (with modest correlation) as witnesses are held (either half-held or fully held).

Under the junta reading, the two estimands that agree most are the two exact ones at the ends of the marginal-contribution family, LOO at the full complement and per-head NIE at the empty context. Note also that here agreement is not accuracy: none of the five orderings is ground truth. If anything is, it is the circuit performance.

Runtime Scoring every candidate set exactly costs $\Theta(2^{|M|})$ forward passes for a pool M, which is infeasible. The per-head grid replaces it with $|M|$ singleton patches, one forward pass per candidate per prompt pair, so it costs $\Theta(|M| \cdot N)$ for N pairs, linear in a pool of 144, 448 or 1024 heads here. JUNTALEARNER instead spends a fixed budget of T steps, each drawing B candidate sets over P prompt pairs, so it costs $\Theta(T \cdot B \cdot P)$ forward-backward passes, independent of $|M|$. A forward-backward pass costs a small constant multiple of a forward one, so the comparison is $|M| \cdot N$ against TBP. The gradient approximations cost $\Theta(m \cdot N)$ for m integration steps, also independent of $|M|$, but their circuits score worse.

Table 2: Distinct trained configurations by model and task.

Task	GPT-2 small	Qwen3-1.7B	Llama-3.1-8B
IOI	51	32	44
SVA	51	36	24
Arithmetic	76	31	32
Induction	51	31	32
Entity tracking	26	29	26
Total	255	159	158

C EXPERIMENTS

In Section 4 we describe the models, tasks, metrics and baselines, and in Table 5 we provide more details about the tasks. Here we discuss how we chose the configurations and how experimental runs were orchestrated.

Choosing configurations. Each training run costs GPU time, so two surrogate models chose the 572 configurations behind Section 4. After each round we fit a Gaussian process to every run so far, one per goal, with the model, the task and the knobs of Table 3 as inputs and the CRS as output. We use BoTorch's `MixedSingleTaskGP`, which pairs a Matérn kernel over the numeric knobs with a kernel over the categorical ones, and pick the next runs by expected improvement. Gradient-boosted trees (CatBoost), fit to the same table, serve as a check on each recommendation. The search covers 24 to 76 configurations per cell (Table 2), and 150 of the 572 runs are seed replicates, so the per-cell best scores in Figure 4 show what tuning buys within a limited search.

One run. Figure 8 follows Llama's default on addition, the ladder 142-71-43-21-7 with no witnesses pinned. The first four rungs each end early, once the drawn sets stop changing. The last rung runs to its step budget while the ordering is still moving, which is what it is for. The candidates' log-probabilities fan out from a flat start, and the separation between the kept candidates and the rest climbs to about 0.3 over the final rung.

Knobs and seeds. Table 3 lists the knobs we varied. Step budgets run from 30 to 90 steps on GPT-2 and Qwen3 and from 30 to 150 on Llama.

Table 3: The knobs varied across the trained configurations.

Knob	Values tried	Meaning
Witness regime	free, half held, held	share of the eligible witness pool pinned
k schedule	ladder, k=1, reverted	decreasing, one flat k, or rising
Ladder shape	top rung 1 to 512, 1 to 7 rungs	compared as a share of the pool
Measurement stage	on, off	measures the leading candidates one at a time in the final steps
Objective	NIE, NIE + PIE, bottleneck	PIE alone is out of scope
Prompt pairs	100, 150, 200	150 on Llama only

The three per-model defaults are GPT-2's `free, 200 pairs`, Qwen3's `free, interior 150, 200 pairs, measured` and Llama's `free, 142-71-43-21-7 ladder`, where the first word names the witness regime. We retrained each default under independent seeds on every task, 5 to 35 times per cell. Each per-cell interval in Section 4 is a 95% t-interval over those seeds, and each pooled interval in Figure 5 is a t-interval over the five per-task seed means. Cells whose first five seeds left the margin indistinguishable from zero got more seeds. That rule depends on the result it measures, so an interval that clears zero on a topped-up cell is weaker evidence than the same interval on a cell replicated to a fixed depth.

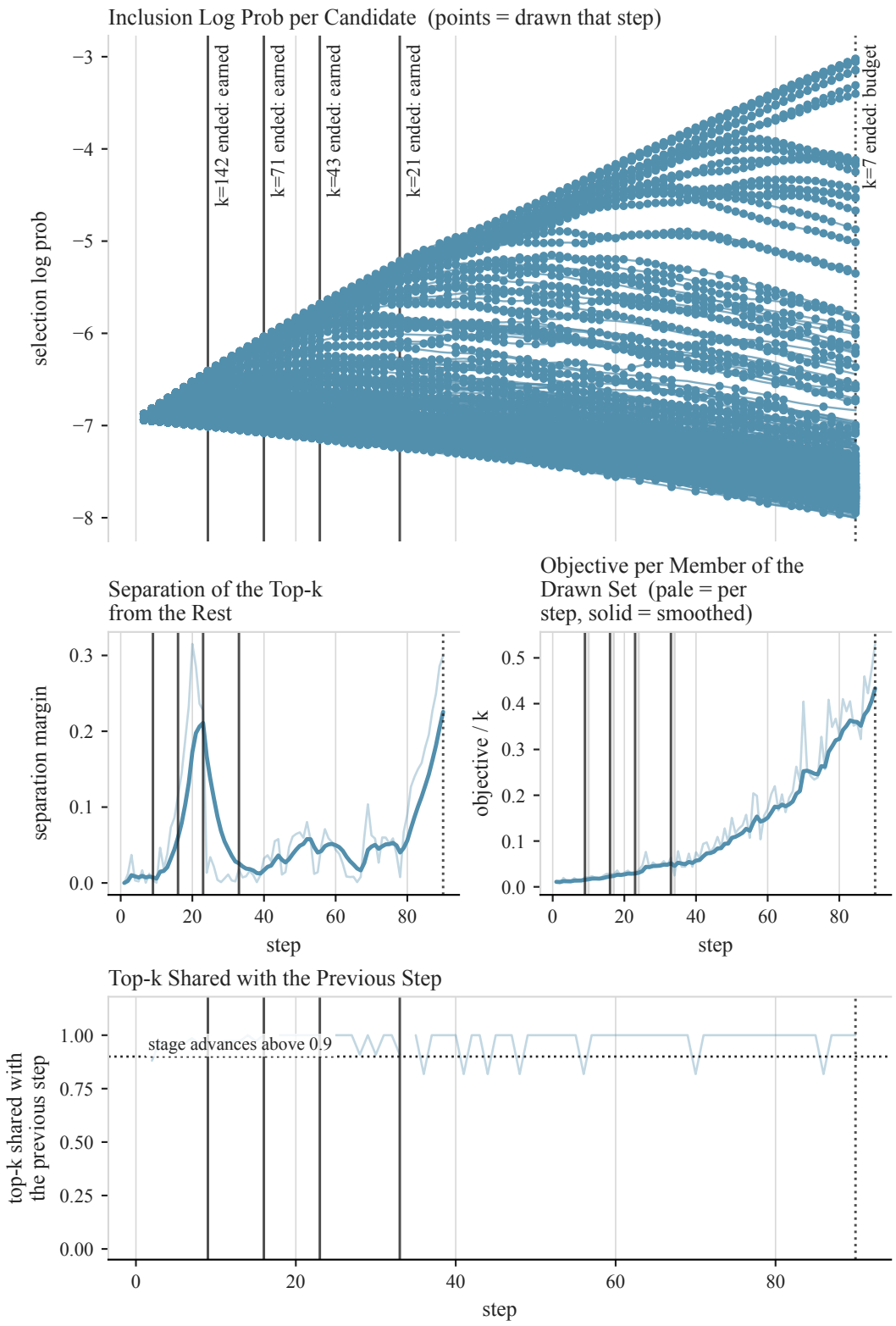


Figure 8: One training run on Llama-3.1-8B: arithmetic, the free witness regime, seed 0, ladder $(142, 71, 43, 21, 7)$. Top: every candidate's inclusion log-prob across steps, a point marking a step that drew it, with vertical rules where a rung ends and why. Bottom left: the separation margin, climbing again once a new rung starts. Bottom right: the mean objective per member of the drawn set, smoothed; a jump at a boundary is the change in k. Bottom: the share of the top k carried over from the previous step, which is the quantity a rung has to hold above 0.9 to end early.

Table 4: Settings per model. The candidate pool and the default ladder describe each model's default; the other rows summarize all of its runs.

Setting	GPT-2 small	Qwen3-1.7B	Llama-3.1-8B
Candidate pool	144 head writes	448	1024
Prompt pairs	100 or 200	100 or 200	100 or 150
Default ladder	20-10-6-3-1	150-67-33-8-1	142-71-43-21-7
Steps per run	30 to 90	30 to 90	30 to 150
Configurations	255	159	158
Seeds per default cell	5 to 35	5 to 10	5 to 10

Table 5: Example factual prompts and the single change that produces each alternative prompt.

Task	Factual prompt → answer	Alternative prompt
IOI	*When Mary and John went to the store, John gave a drink to* → Mary	Second subject mention replaced by a third name
SVA	*The friends* → have	Subject number flipped (*The friend* → has)
Arithmetic	*Q: How much is 27 plus 64? A:* → 91	Both operands resampled
Induction	A random word block, repeated: *oak map … rat hat oak map … rat* → hat	Boundary word changed (*hat* → *cow*)
Entity tracking	*The apple is in Box C. The key is in Box A. … Box C contains the* → apple	Same box queried, different object placed in it

D RESULTS IN DETAIL

Figures 9 to 11 show, for each of the fifteen cells, the best-scoring trained configuration against all eight baselines. A reader who distrusts the aggregation can check here whether a higher CRS corresponds to a walk they would have preferred by eye. The winner is reselected per task and goal.

Figures 12 to 14 put several settings on one grid per model, with CRS bars. The difference is that we chose one default configuration per model, without optimizing for task-specific performance.

Moving to uncertainty, Figure 15 takes the per-model default, the single configuration Section 4 selects for each model, and retrains it under several seeds on each of that model's five tasks. each with a 95% interval. To illustrate that defaults that work decently across tasks can be beaten by goal and task optimized ones, Figure 16 replaces that single defaul t with best configuration kill-wise on that specific (model, task).

Table 2 in Appendix C counts the experiments actually run per (model, task) pair, which shows that we did not extensively over-optimize per-task configurations.

Every contrast in Figure 17 and in Figure 19 is built using a GP trained on the experimetnal results. For a knob k with levels a and b, take model m's own recorded training runs $\mathcal{D}_m$. For each row $x_i \in \mathcal{D}_m$, build two copies agreeing on every covariate except k, stack all $2|\mathcal{D}_m|$ of them into one input X, and draw $S = 300$ samples jointly from a Gaussian process fitted to that goal. Figure 19 repeats the grid with the task fixed as well. Figure 18 ranks every knob by how much shuffling its column hurts the fitted model.

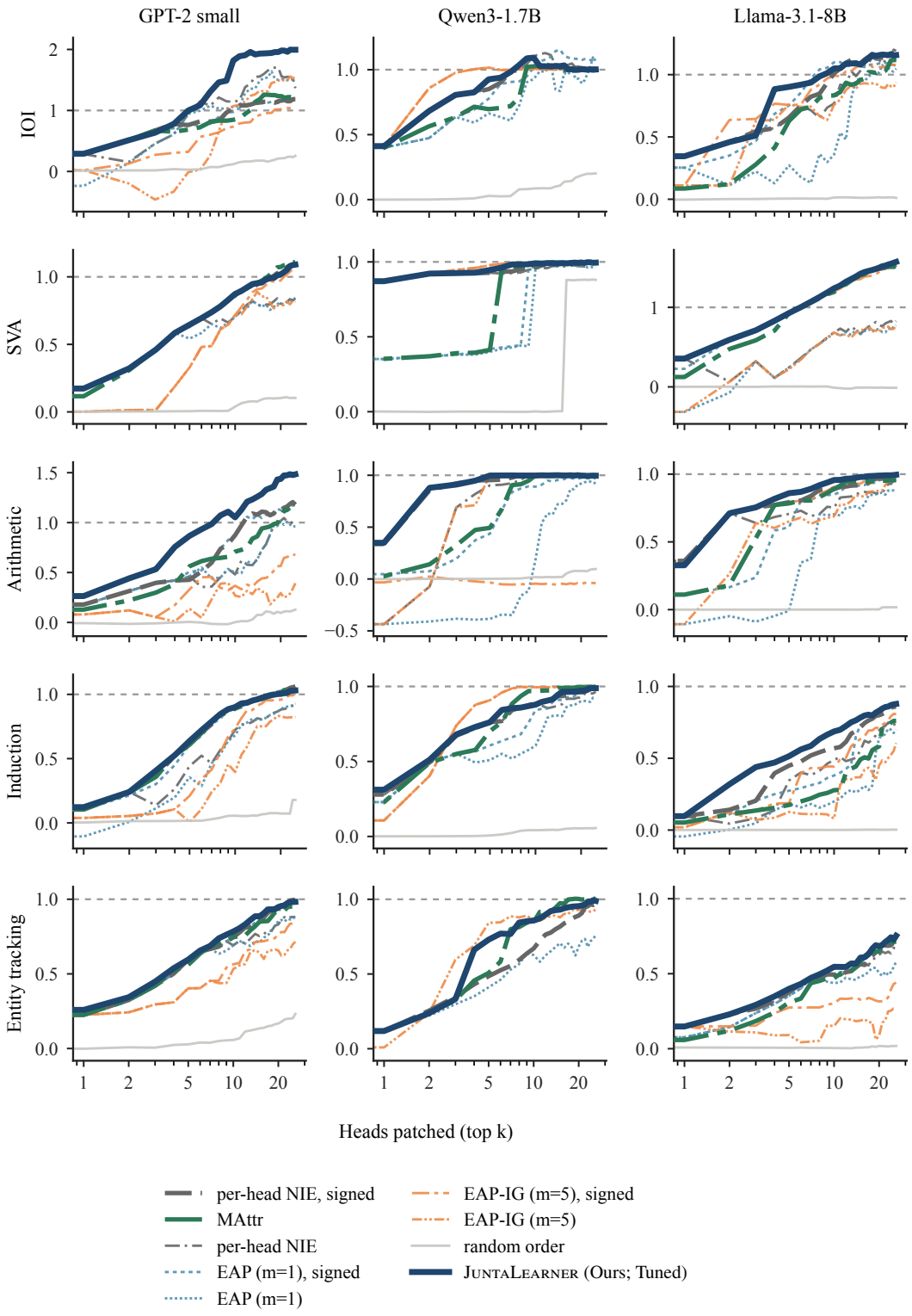


Figure 9: Kill curves for each cell's best-scoring trained configuration (best configuration, for brevity) against all eight baselines. The x axis is the number of attention heads patched, log-scaled.

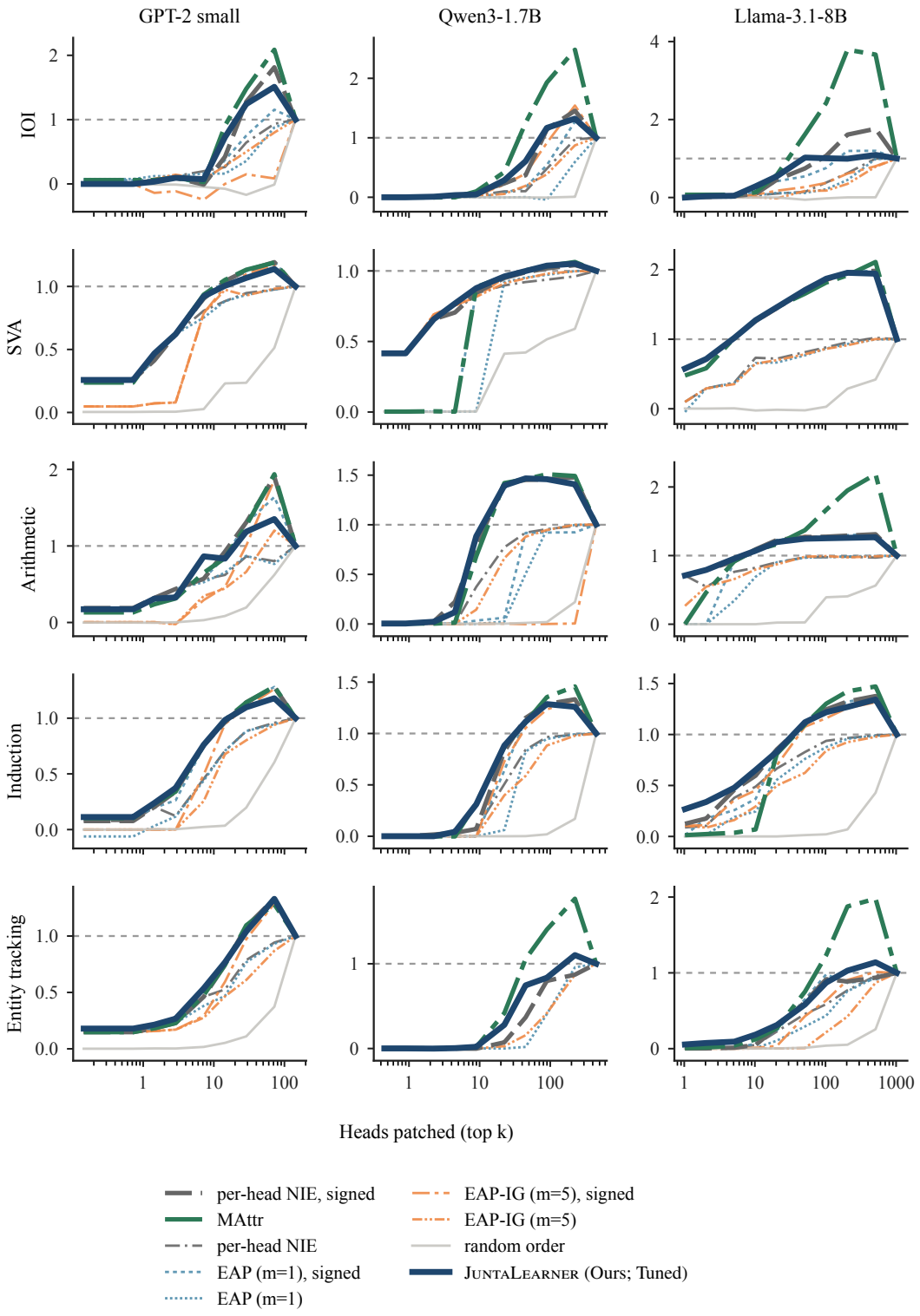


Figure 10: Faithfulness curves for each cell's best configuration, selected on S_{faith} and so not always the same configuration as Figure 9.

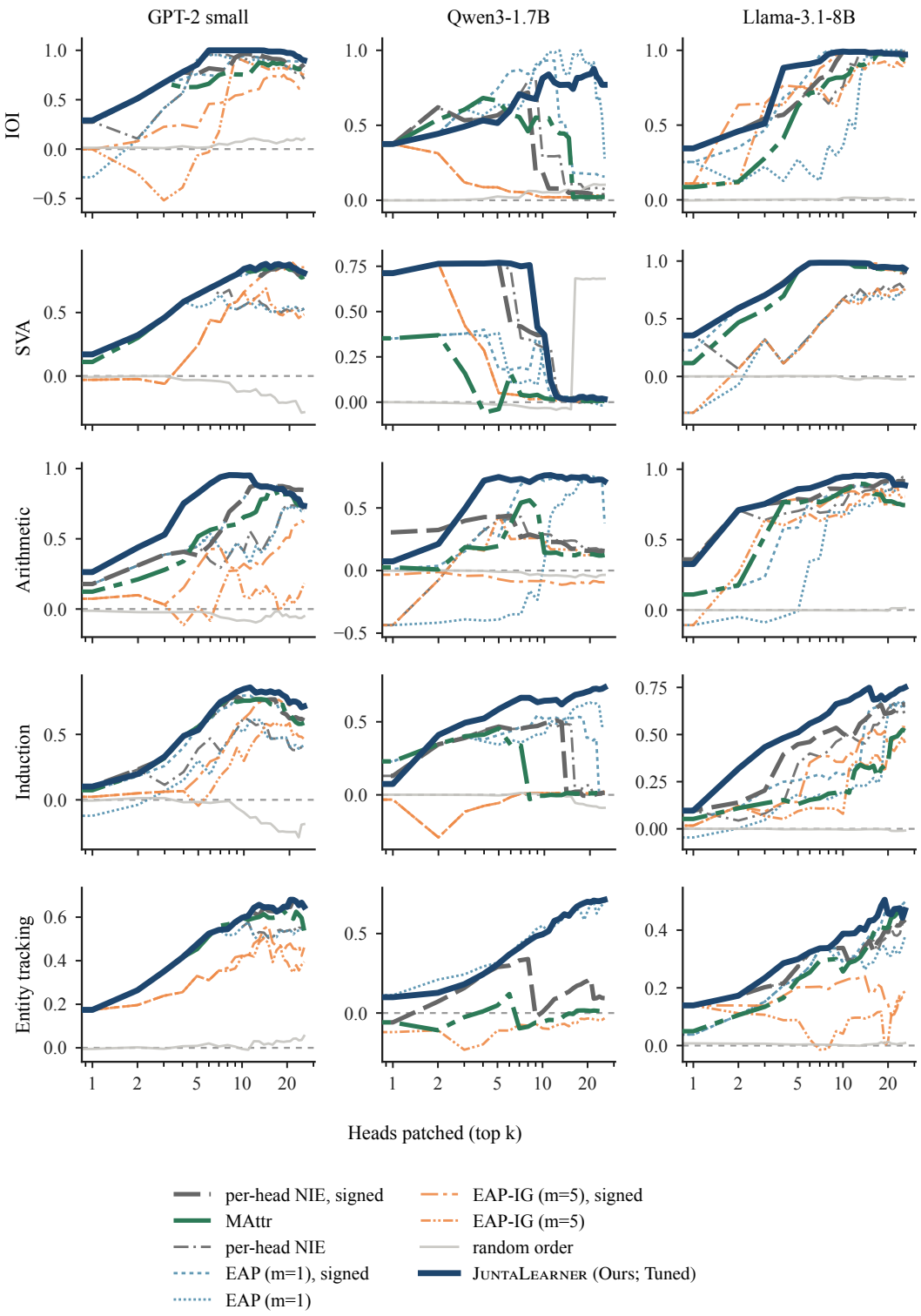


Figure 11: Net curves for each cell's own best configuration. Net regularly picks a different winner from kill and faithfulness.

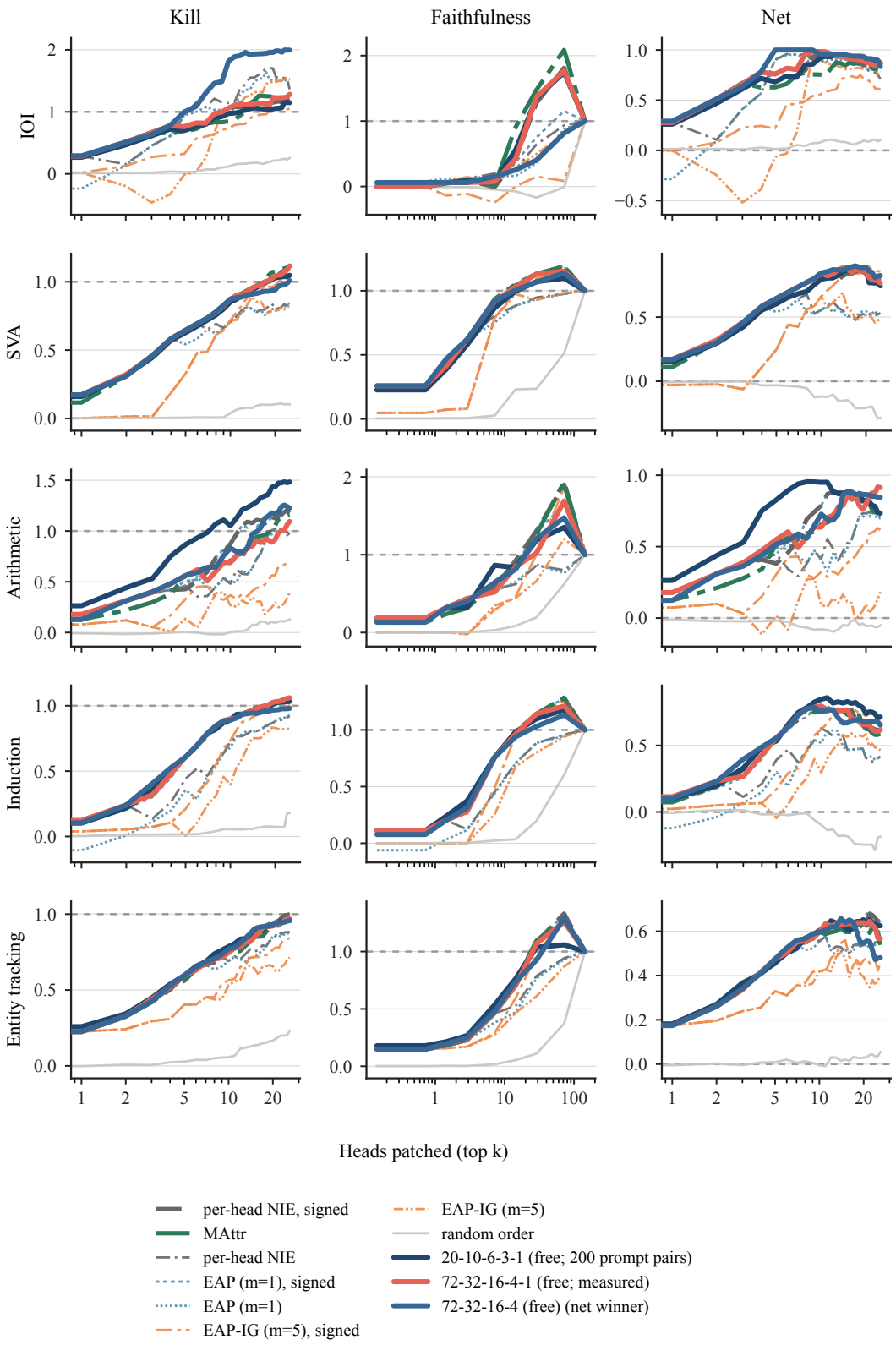


Figure 12: GPT-2 small: three best settings against the baselines on all three goals, with the CRS ranking at the right of each row.

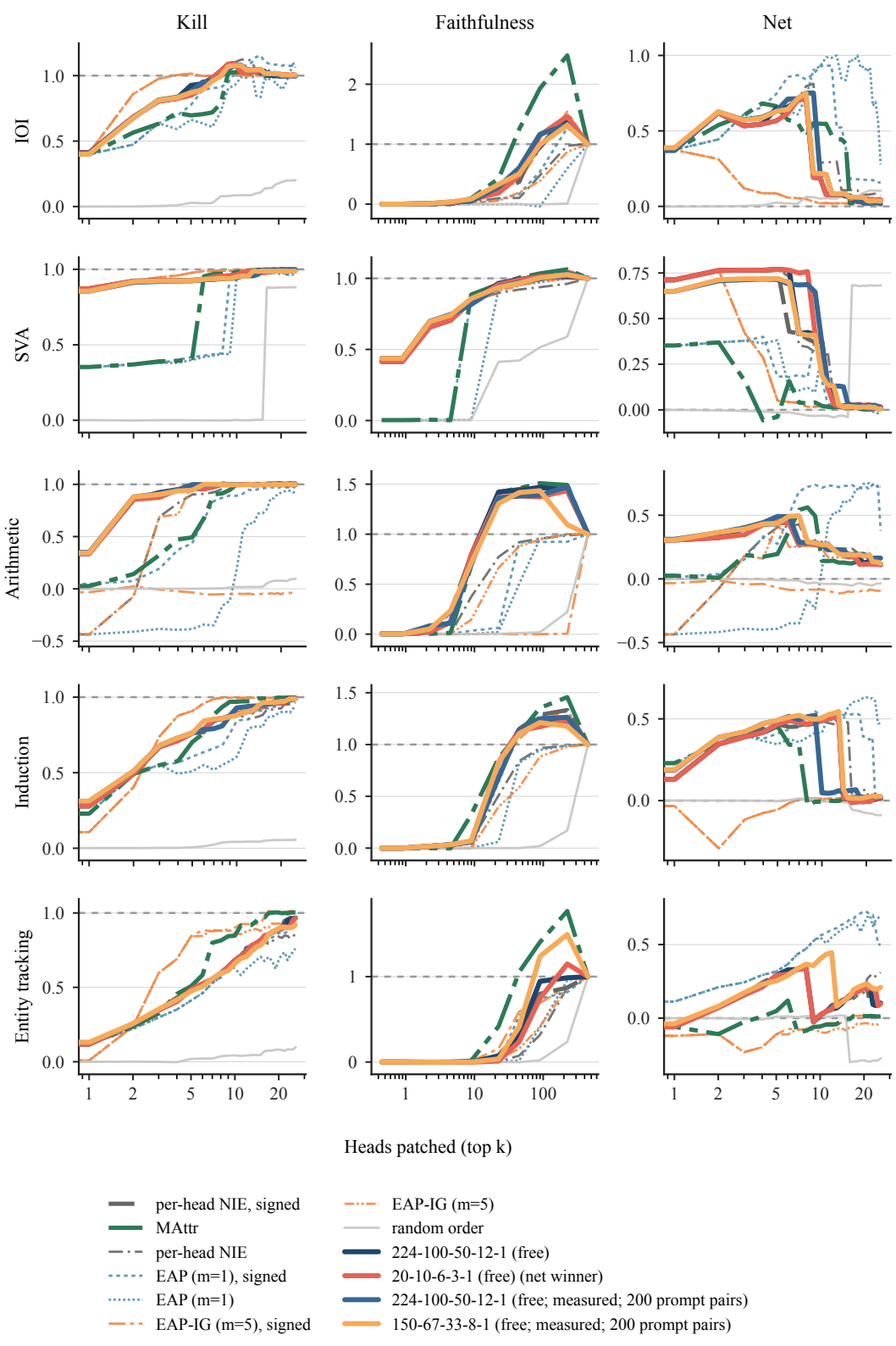


Figure 13: Qwen3-1.7B, drawn as Figure 12. Its ladder shapes score within a few thousandths of each other on kill.

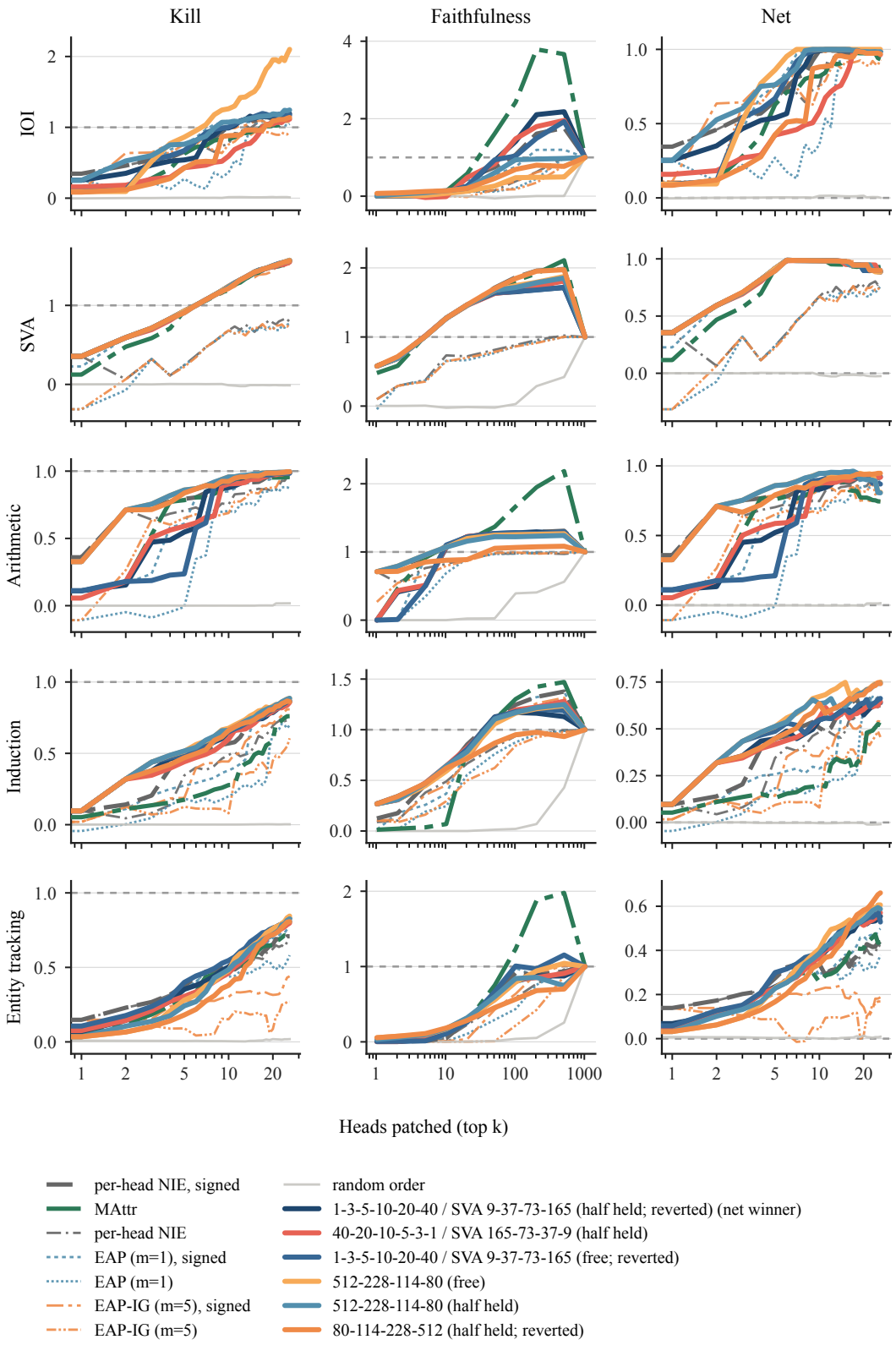


Figure 14: Llama-3.1-8B, drawn as Figure 12.

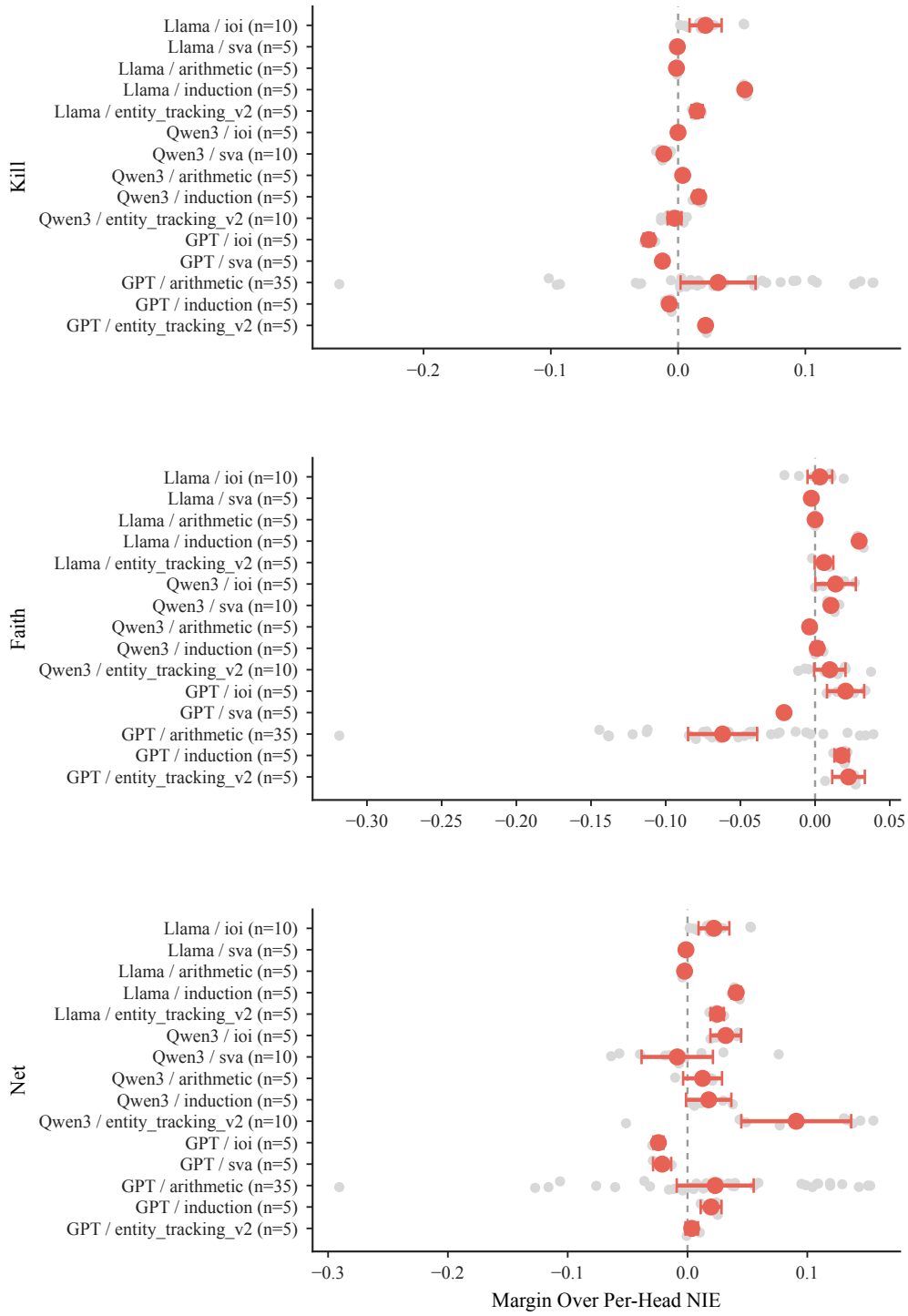


Figure 15: Per-cell margins over signed per-head NIE for model-level default configurations.

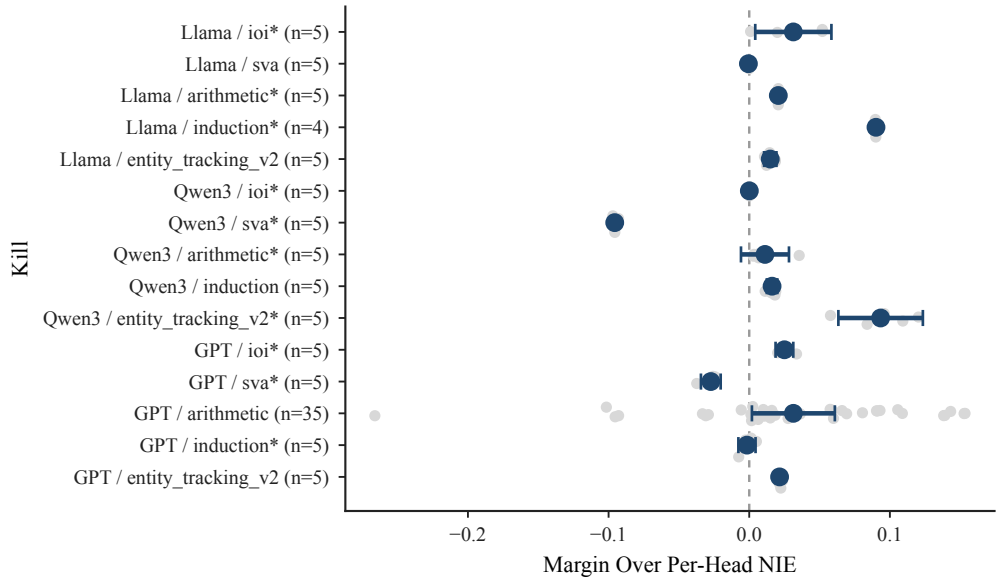


Figure 16: Kill margins over signed per-head NIE for each cell's own winner, drawn as Figure 15.

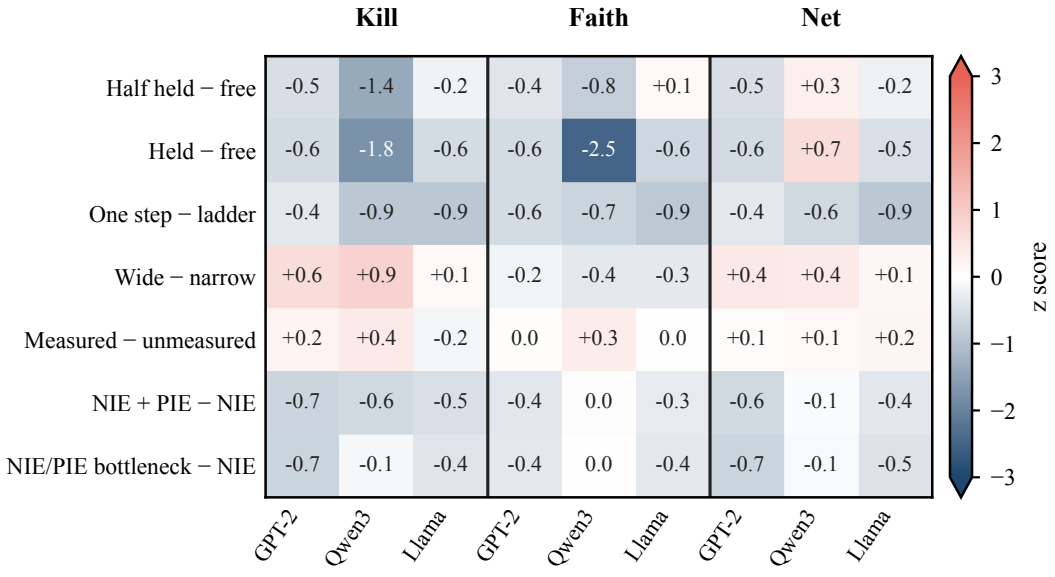


Figure 17: GP-estimated effects of each knob, grouped by goal and model. Rows compare each setting while holding other settings at their observed values. Cells give the posterior mean difference divided by its standard deviation.

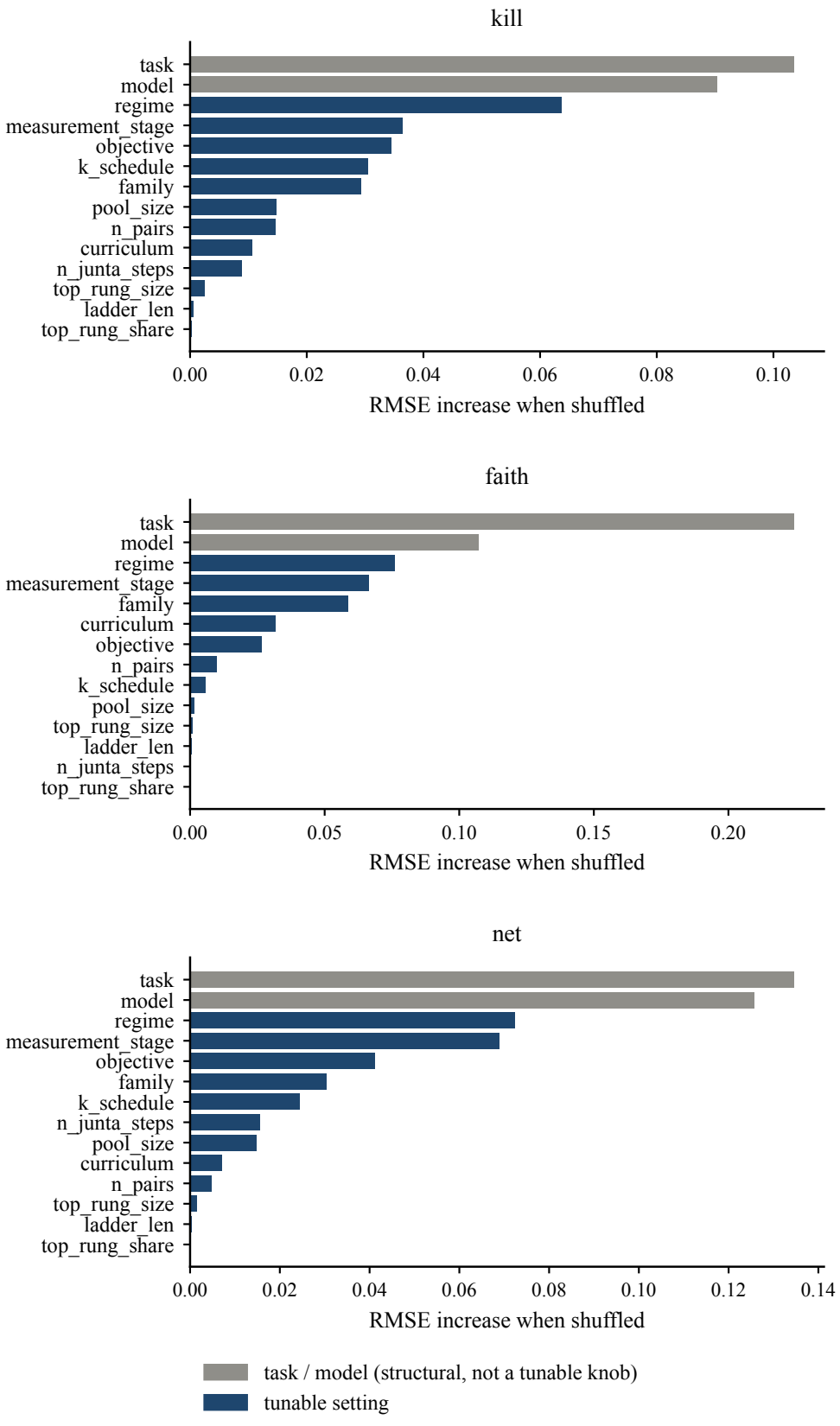


Figure 18: Permutation importance over the fitted model: the increase in error when one column is shuffled.

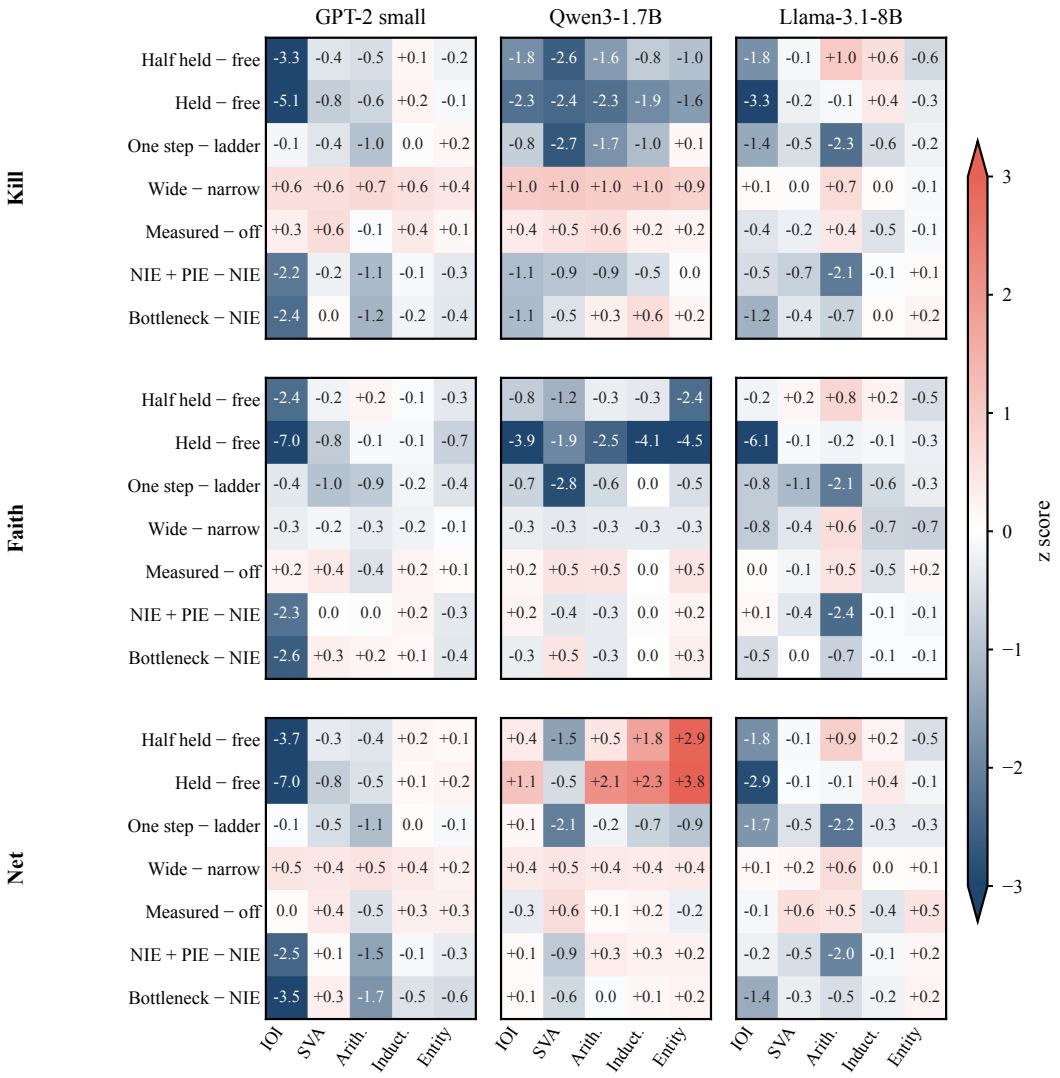


Figure 19: Task-specific knob contrasts across the three goals (rows) and models (columns). Each cell is the mean GP posterior difference divided by its standard deviation. The signed color scale is shared across panels and clipped at ± 3; labels denote the first setting minus the second.

E MIB BASELINE AND COMPARISON TO MATTR

Throughout the paper we summarize curves by CRS, for the reasons given in Section 4. Moreover, a serious baseline to compare to, MAttr (Arora et al., 2026), was published just days before our submission, so this appendix uses the metrics to inform our discussion of our evaluations against this baseline. The first step in including MAttr was to apply the same evaluations to it as we did to all other methods, including ours, in the paper. This, however, might face the following two objections. (1) We used CRS, which is, regrettably, not a metric commonly used in mechanistic interpretability, and so it is not clear whether this is not just another case of p-hacking. (2) We spent some time developing with those tasks in mind, while we only ran MAttr with its default settings, so the comparison might not be fully fair.

Let's address those, as doing so will not only allow us to better understand the MAttr results, but also strengths and weaknesses of the evaluation methods we are proposing and their incumbents, CPR and CMD on faithfulness.

We address (1) first. **On our tasks, our methods lead on CRS and $1 - \mathrm{CMD}$, and MAttr leads on faithfulness CPR** (Figure 20); summed over all three goals, our tuned configuration leads on CPR as well. MAttr's profile across the three summaries is uneven: **on faithfulness it attains the highest CPR and the lowest $1 - \mathrm{CMD}$, with a CRS close to ours**. Two properties of CPR account for this. It gives substantial weight to the right-hand part of the curve, reached after most of the relevant components have been recovered, and so rewards late recovery as much as early detection; and it credits overshoot. Figure 21 shows both effects: MAttr's mean faithfulness curve starts below the others at small shares of the pool (left), and **most of MAttr's CPR advantage comes from overshoot** (right).

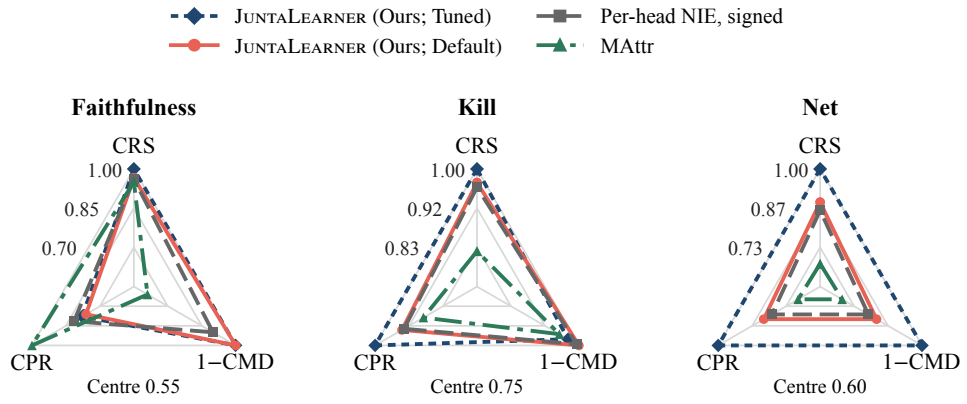


Figure 20: The fifteen cells of the main experiments: *ours, tuned*, *ours, default*, signed per-head NIE and MAttr with its authors' published MIB recipe. Note that for readability each triangle has its own scale and that values between goals are not directly comparable. *Ours, tuned* is chosen per cell and goal by CRS, as in Table 1, and scored under all three summaries. Each axis shows a method's summed score relative to the best, so outward is better on every axis; CMD appears as $1 - \mathrm{CMD}$.

.

Now, let us turn to (2). MAttr seems to have been developed and evaluated mostly with the MIB's circuit track (Mueller et al., 2025). While we had our independent reasons to choose our tasks, the limited time allows us to see how our *untuned* methods compare to MAttr on those tasks.[6] **On MIB's public test split, our untuned run is close to a tie with MAttr's circuits on CRS summed over the**

[6]MIB's circuit track pairs four models with six tasks in twelve cells, scored at the node level, where a circuit is a set of attention heads, MLPs and the input embedding; its Qwen model is Qwen2.5-0.5B. Our MIB scoring for this paper comes from the public test split, in which we follow Arora et al. (2026). As for settings, our only choice on MIB data was the training objective, NIE, made on MIB validation from pilot runs of four objectives on GPT-2 IOI; a few further objective and configuration variants on the Qwen and Llama IOI and MCQA cells served only as sensitivity checks. Every score reported here comes from one training seed per cell. Faithfulness comes from MIB's own evaluator. Kill and net walk each ranking's top attention heads as in the

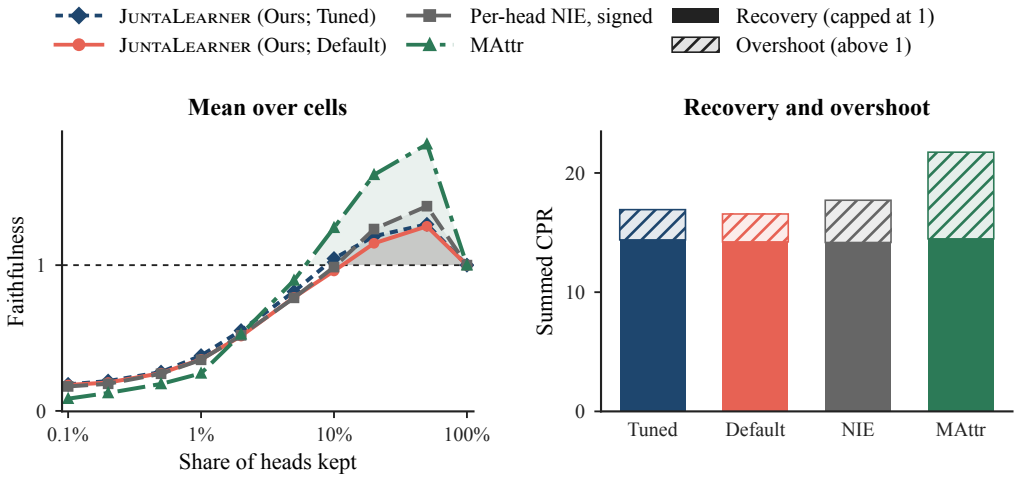


Figure 21: Faithfulness for fifteen cells of the main experiments. Left: each method's faithfulness curve, averaged over cells; shading marks where it rises above 1. Right: each method's faithfulness CPR, split in two components: area under the curve capped at 1 with area above one stacked 13above.

three goals, ahead on $1 - \mathrm{CMD}$ **and behind on CPR**. As we also care about the other evaluation axes, we include also triangles that correspond to them (Figure 22). **On kill and net, summed over cells, our untuned defaults beat MAttr on all three summaries.** Moreover, the CPR dominance for MAttr comes to a large extent from overshoot here too. **Most of MAttr's CPR lead on MIB comes from overshoot.** (Figure 23). That being said, notice how the mean MAttr for faithfulness here climbs a bit faster than ours, and so **CRS rightly prefers MAttr's released circuits over our untuned default for faithfulness on MIB**.

MAttr is a strong method, and the two methods are trained in opposite directions, which may explain some of the uneven picture. MAttr's released circuits are optimized for what MIB faithfulness explicitly scores: to keep a set of components and patch everything else. Ours are trained by patching the set itself, which is closer to what is scored by kill. Sums over cells can hide how these differences play out on individual cells, so Figures 24 and 25 provide every curve behind them, one row per cell (MIB's faithfulness, faithfulness over heads, kill and net), for the reader's spiritual elevation.

main experiments, on up to 200 test pairs (as few as 17 on MCQA and ARC), with MLPs left out, since on GPT-2 IOI patching the first-layer MLP alone removes every task and saturates both curves.

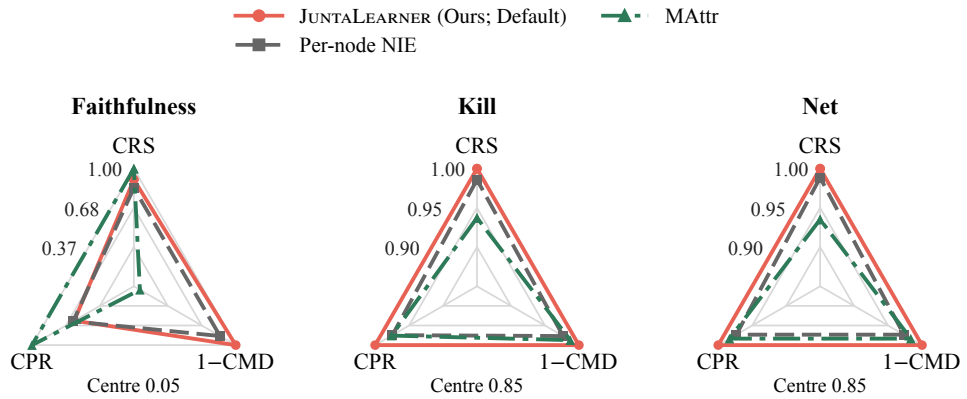


Figure 22: The twelve MIB cells, public test split: *ours, default* in one untuned pass, per-node NIE and MAttr's released circuits, drawn as Figure 20, with faithfulness from MIB's evaluator. CMD appears as $1 - \text{CMD}$.

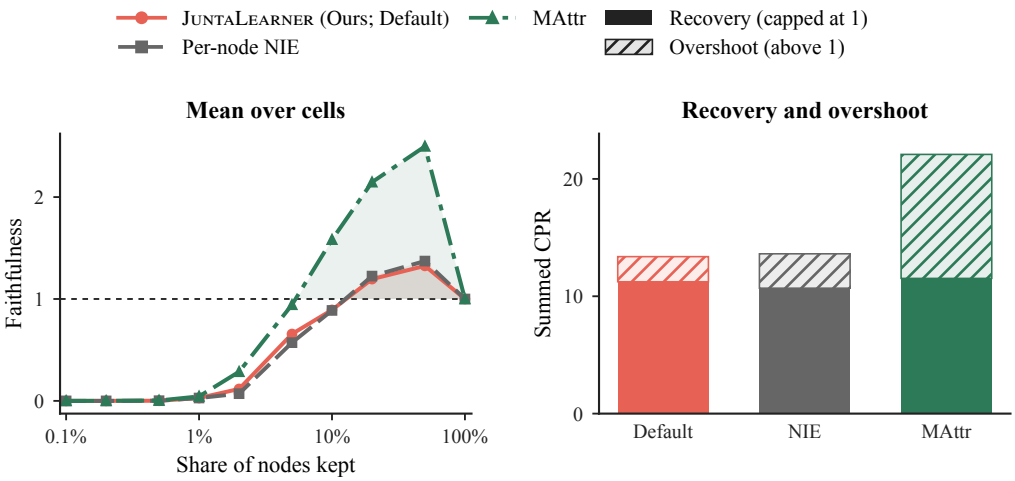


Figure 23: Twelve MIB cells, drawn and computed as Figure 21.

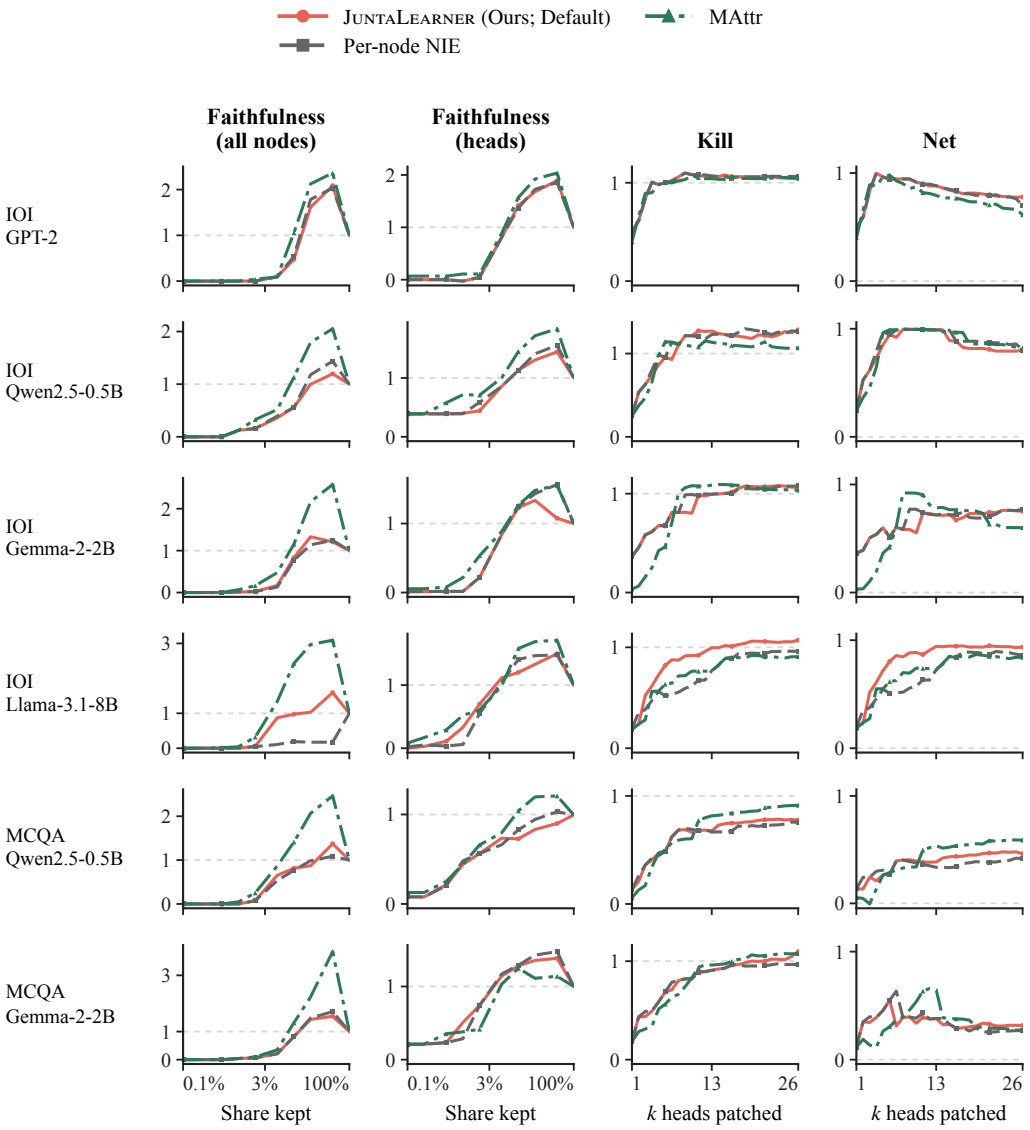


Figure 24: MIB public test, the four IOI cells and two of the three MCQA cells. Columns: MIB faithfulness over all nodes and faithfulness over heads (share of the pool kept), then kill and net (top k heads patched). Dashed grey marks $f = 1$ and kill $d = 1$ (d is the share of the full-pool effect removed), the ideal values, and net $= 0$, where a circuit's damage to the target equals its mean damage to the controls; the ideal net is 1.

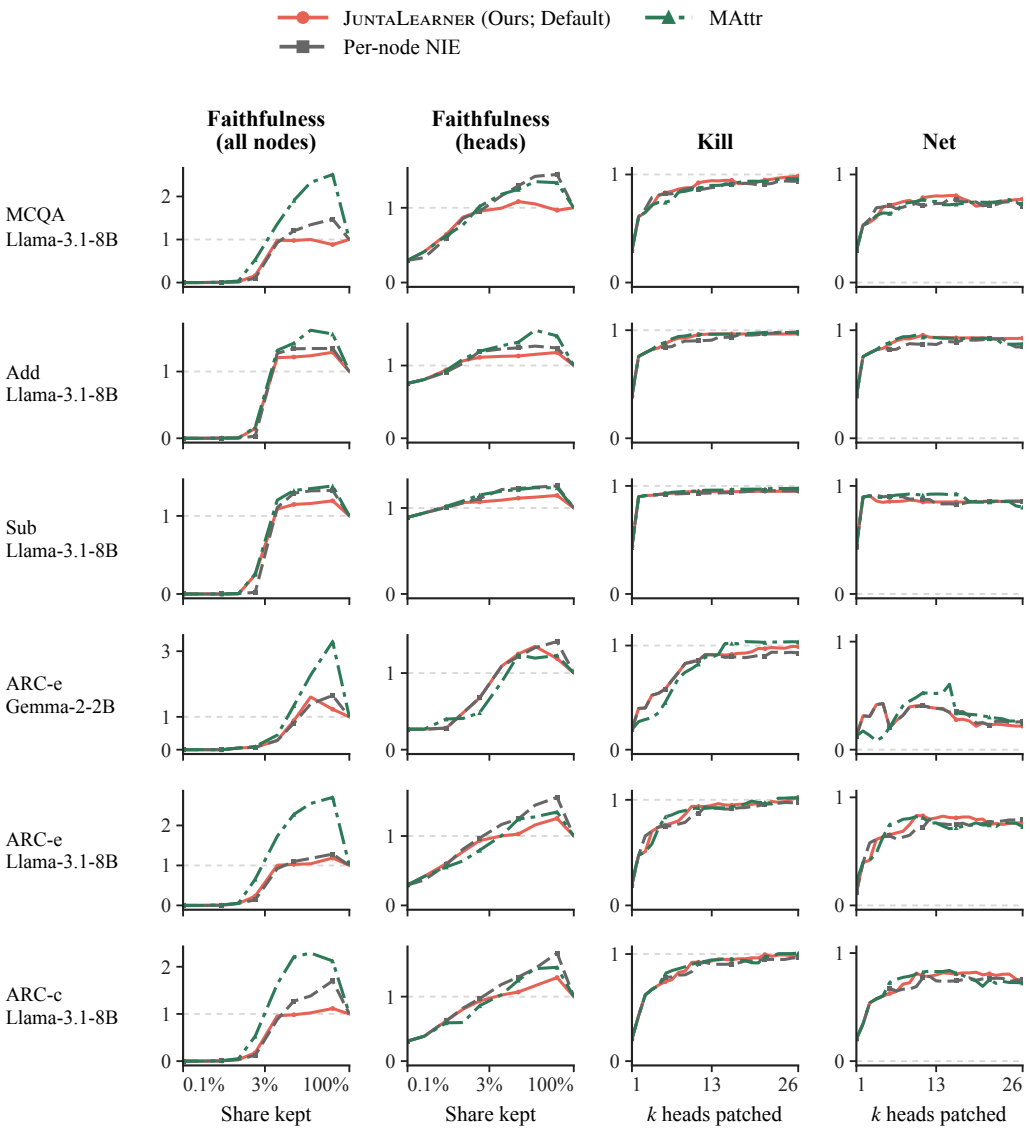


Figure 25: MIB public test, the remaining cells, drawn as Figure 24.