GUIDING ATTENTION CIRCUIT DISCOVERY WITH REASONING OUTCOMES

Anonymous authors
Paper under double-blind review

ABSTRACT

We adapt attention circuit discovery to reasoning models by learning which attention connections between sentences in a reasoning prefix must be preserved to promote a given reasoning outcome. Because interventions can change the distribution of reasoning continuations, we train masks on continuations sampled from the unmasked model and test them on held-out unmasked continuations and on fresh continuations generated under the learned mask. We consider three objectives—preserving answer distributions, increasing correct-answer probability, and shortening reasoning—and find that our approach is more successful than a local connection baseline that scores connections by their effects on subsequent token predictions. We then investigate the learned circuits and find that retained connections show no consistent preferences for different types of reasoning steps. Moreover, different training seeds also produce distinct circuits with similar performance across sparsity levels. Together, these findings suggest that language model "reasoning" lacks the crisp, systematic structure evoked by the word.

1 INTRODUCTION

Causal analysis of internal representations has made substantial progress, providing tools to identify the computations underlying neural network behavior and test hypotheses about them through targeted interventions (Bau et al., 2019; Conmy et al., 2023; Elazar et al., 2021; Finlayson et al., 2021; Geiger et al., 2024; 2025; Meng et al., 2022; Vig et al., 2020; Wang et al., 2023). However, many of these methods are developed and evaluated in constrained settings with templated inputs, fixed token sequences, and narrowly specified outputs. Reasoning models pose a different problem because their computations unfold over long, stochastically generated continuations. This means an intervention can change both the answer and the reasoning that leads to it. Adapting causal analysis to this setting requires specifying which outcomes an intervention should produce and which distribution of continuations should be used to assess it.

Studies of generation and reasoning show that alternative continuations from the same prefix can yield substantially different outcomes (Bigelow et al., 2024; Macar et al., 2025; Zur et al., 2025). A mask trained on one recorded continuation may therefore fail on other continuations from the same prefix. Even a circuit that performs well across several fixed traces may fail when it changes the model's subsequent generation. We argue that circuit discovery for reasoning should be guided by reasoning outcomes, with explicit evaluation of whether the desired effects persist across continuations and under the distribution induced by the intervention.

To this end, we adapt *desiderata-based masking*, a family of methods that learns differentiable masks by optimizing objectives on their intervention effects (Cao et al., 2021; Davies et al., 2023; De Cao et al., 2020; 2021; Prakash et al., 2025). We learn a sparse mask over attention connections between sentences in a reasoning prefix, keeping model parameters fixed. The circuit is the subgraph of the computation graph whose connections remain intact. Each circuit is specific to a prefix and an objective, and is trained on one or more continuations sampled from the unmasked model. The objectives can specify preservation of the model's behavior or promote a different outcome, such as a change in answer probabilities or reasoning length.

We evaluate the learned circuits using two complementary protocols. *Fixed-suffix evaluation* processes held-out continuations from the unmasked model with the learned mask applied, testing whether its effects generalize beyond the training suffixes. *Generated-suffix evaluation* samples fresh

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.

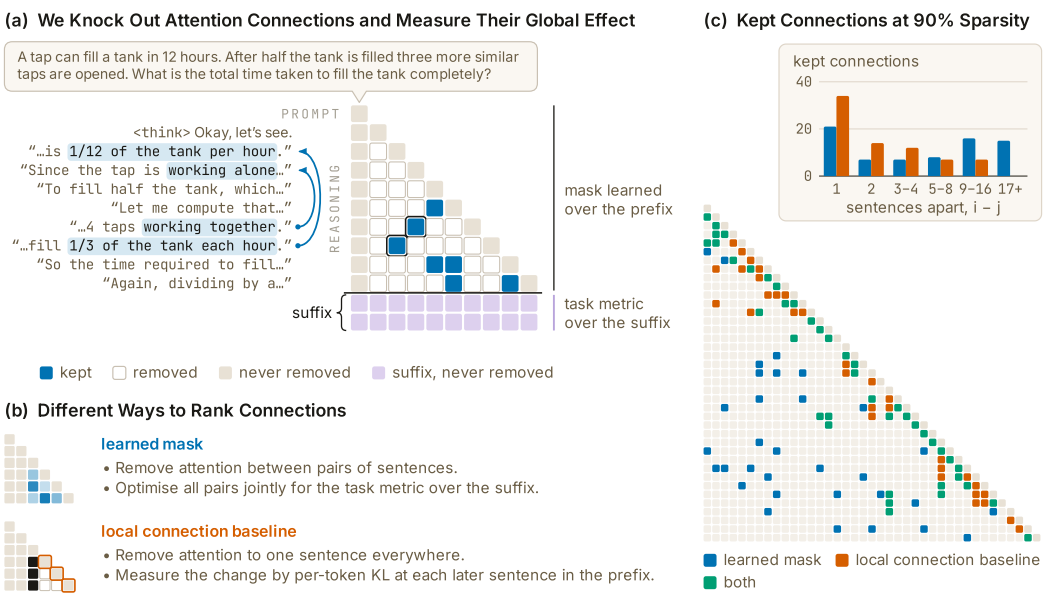


Figure 1: Attention circuit discovery from reasoning outcomes. (a) Each cell (i, j) represents attention from sentence S_i to sentence S_j across all layers and heads. The mask acts within the prefix x. Within-sentence attention and connections involving the prompt or the <think> segment remain active. (b) We train candidate masks $\widetilde{M}$ using a reasoning outcome loss on training suffixes and a sparsity penalty. The trained mask M retains the k connections with the largest learned activation probabilities. The local connection baseline retains the k pairs with the largest scores c_{ij}, which measure the change in next-token distributions at S_i when attention to S_j is removed. (c) Learned and baseline masks for one GPQA prefix at 90% sparsity over the eligible connections $\mathcal{E}$.

continuations from the masked model and measures the resulting reasoning outcomes. Both keep the prefix and mask fixed. The masks are trained on a fixed set of rollouts and these two protocols evaluate different kinds of generalization from that training data.

Our primary contributions are as follows.

- We formulate attention circuit discovery for reasoning as desiderata-based masking with objectives defined on reasoning outcomes. We develop a training and evaluation setup that distinguishes performance on fixed off-policy reasoning sequences from on-policy reasoning generated by a masked model.
- We study attention circuits across three different outcome metrics: preserving answer distributions, increasing probability of the correct answer, and reducing reasoning length. We compare our method to a local connection baseline (Bogdan et al., 2025) that ranks connections by their effects on subsequent token predictions, and our method is best across all three experiments.
- Our analysis of the learned circuits suggests that reasoning relies on distributed and redundant information. Retained connections span positions throughout the reasoning prefix, with local connectivity most relevant to preserving the answer distribution. These connections show no consistent preference for particular categories of reasoning steps across objectives. Different training seeds also produce distinct circuits with similar performance, suggesting that multiple sets of connections can support the same reasoning outcome.

2 Methodology and Experimental Setup

We learn sparse attention circuits by optimizing objectives defined on reasoning outcomes. For each reasoning prefix and objective, we find a circuit (sparse set of attention connections) on one

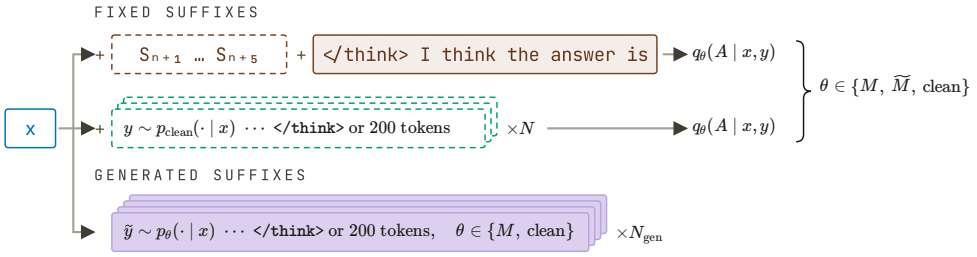


Figure 2: Training and evaluation for a fixed reasoning prefix x. Fixed suffixes: We append a set of fixed suffixes ($\mathcal{D}_{\text{train}}$) sampled from the unmasked (clean) model and read the answer distribution q_θ under a candidate mask ($\widetilde{M}$) during training, or a binary mask M during evaluation. The top row shows a training example with $|\mathcal{D}_{\text{train}}| = 1$, used as a training objective. The middle row shows another version where we generate N rollouts and use them for training (D_{train}) or hold out for evaluation (D_{test}). Generated Suffixes (bottom row): Sample N_{gen} rollouts each using a binary masked model as well as the unmasked model. We use these during evaluation to understand global properties of the sampled rollouts.

or more continuations of that prefix. We evaluate the circuit on additional continuations using fixed (off-policy) and generated (on-policy) suffixes.

2.1 Masking Connections Between Reasoning Sentences

Prefixes and suffixes. Let x be a prefix that contains a prompt and the reasoning generated up to a chosen analysis point. We divide the reasoning in x into sentences, with S_i denoting the set of token positions in sentence i, for $i = 1, \ldots, n$. Let Y be the random variable for suffixes that continue x, and let y denote a particular suffix. A complete suffix contains the remaining reasoning and final answer; each experiment specifies where suffix sampling stops. We write $p_{\text{clean}}(Y \mid x)$ for the distribution over suffixes under the unmasked model, with $p_{\text{clean}}(y \mid x)$ denoting the probability of the token sequence y. We sample every suffix at temperature 0.7 unless stated otherwise, so $p_{\text{clean}}(Y \mid x)$ and the masked distributions defined below denote sampling at that temperature. However, the answer and next-token distributions that we read from a forward pass, without sampling, are the softmax of the unscaled logits.

Attention circuits. Let $\mathcal{E} \subseteq \{(i, j) : 1 \leq j < i \leq n\}$ index the sentence-to-sentence connections eligible for removal, where (i, j) denotes attention from sentence i to sentence j. Unless stated otherwise, we do not ablate over a sentence's attention to itself ($i = j$) as it throws models out of distribution.

A candidate mask $\widetilde{M} \in [0, 1]^{\mathcal{E}}$ assigns a value $\widetilde{M}_{ij}$ to each eligible pair. Let z_{qk} denote the attention logit for the token at position q attending to the token at position k. At every layer and head, we apply the same mask before softmax:

$$z_{qk} \leftarrow z_{qk} + \log \widetilde{M}_{ij}, \qquad q \in S_i, \quad k \in S_j, \quad (i, j) \in \mathcal{E}. \tag{1}$$

A zero mask value removes the connection through the convention $\log 0 = -\infty$. During training, we bound the added log value below by $\log 10^{-30}$, after which softmax renormalizes attention. We apply the final binary mask $M \in \{0, 1\}^{\mathcal{E}}$ in the same way.

Attention within each sentence and connections involving the prompt remain active. The mask applies only within the prefix, so suffix tokens retain ordinary causal attention to the prefix and earlier suffix tokens. We write $p_{\widetilde{M}}(Y \mid x)$ and $p_M(Y \mid x)$ for the resulting suffix distributions. Model parameters remain fixed.

2.2 Desiderata-Based Masking

We apply desiderata-based masking to learn a differentiable mask over attention connections between sentences in the reasoning chain, trained to minimize a task-specific loss function.

Reasoning outcomes and losses. A reasoning outcome is a measurable property of a continuation, such as its answer distribution or length. To compute the task loss $\ell(\widetilde{M}; x, y)$, we supply the sampled suffix y to the masked model and use the resulting predictions. The suffix tokens stay fixed when computing loss. Each experiment has its own loss and evaluation metrics.

Training. We sample a training set $\mathcal{D}_{\text{train}}$ of suffixes from $p_{\text{clean}}(Y \mid x)$, with 32 suffixes for each x for answer preservation and 12 suffixes per x for shortening. By contrast, the correct-answer task trains on a single suffix for each prefix, the five sentences that follow x in the original trace. For target sparsity s, the circuit retains $k = \text{round}((1-s)|\mathcal{E}|)$ connections. Each eligible connection (i, j) has a positive parameter α_{ij}. We optimize its logarithm, $\phi_{ij} = \log \alpha_{ij}$, initialized as specified in Appendix B.3, and collect these parameters in $\phi = (\phi_{ij})_{(i,j)\in\mathcal{E}}$. The parameter ϕ_{ij} controls the hard-concrete distribution from which we sample $\widetilde{M}_{ij}$; this distribution permits intermediate mask values during training and assigns positive probability to both 0 and 1 (Cao et al., 2021; Louizos et al., 2018). We write Q_ϕ for the resulting distribution over candidate masks. Appendix B.3 gives the sampling equations. Let $\pi_{ij}(\phi) = \Pr_{\widetilde{M}\sim Q_\phi}(\widetilde{M}_{ij} > 0)$ be the probability that connection (i, j) is active in a sampled mask. The training objective is

$$\mathcal{L}_{\text{train}}(\phi; x) = \mathbb{E}_{\widetilde{M}\sim Q_\phi}\left[\frac{1}{|\mathcal{D}_{\text{train}}|}\sum_{y\in\mathcal{D}_{\text{train}}} \ell(\widetilde{M}; x, y)\right] + \frac{\lambda}{|\mathcal{E}|}\left(\sum_{(i,j)\in\mathcal{E}} \pi_{ij}(\phi) - k\right)^2. \quad (2)$$

Here $\lambda \geq 0$ weights the sparsity penalty. Each training step estimates the expectation using independently sampled masks and averages their task losses. For answer preservation, we use four masks per step, and for the correct-answer and shortening tasks, we use one mask per step. We use Adam for gradient descent, with a separate gradient update for the sparsity penalty. Appendix B.3 gives the optimizer settings and the schedule for λ. We obtain a binary mask M that retains the top k connections with the largest learned activation probabilities and removes the rest. We learn one mask for each combination of prefix, objective, and target sparsity.

2.3 Evaluation

Figure Figure 2 shows how we evaluate each mask using two kinds of reasoning data: *Fixed Suffix Evaluations* use held-out off-policy rollouts sampled from the unmasked model, while *Generated Suffix Evaluations* compare on-policy rollouts generated by the masked model with those generated by the base model. We also consider a local connection baseline, adapted from Bogdan et al. (2025).

Local connection baseline. We compare with a baseline adapted from Bogdan et al. (2025). Its connection scores depend only on the fixed prefix x. Let p_{-j} be the model with attention from prefix tokens to sentence j removed in every layer and head. We denote V_{t+1} as the random variable for the next token that the model predicts at position t, whose values range over the model's vocabulary, and write $x_{\leq t}$ for the prefix tokens up to and including position t. The score for a pair (i, j) is the average next-token KL divergence over the positions of sentence i:

$$c_{ij} = \frac{1}{|S_i|}\sum_{t\in S_i} D_{\text{KL}}(p_{\text{clean}}(V_{t+1} \mid x_{\leq t}) \,\|\, p_{-j}(V_{t+1} \mid x_{\leq t})), \qquad j < i. \quad (3)$$

Bogdan et al. (2025) average the logarithm of the token-level KL divergences and subtract, for each sentence i, the mean score over all earlier sentences. We average the KL divergences themselves and skip this subtraction, so the scores of all pairs share one scale for the top-k selection. The baseline retains the k highest-scoring pairs in $\mathcal{E}$. We also compare with masks that retain k pairs chosen uniformly at random. Once selected, baseline masks use the same test suffixes and evaluation protocols as the learned masks, with the same connection budget.

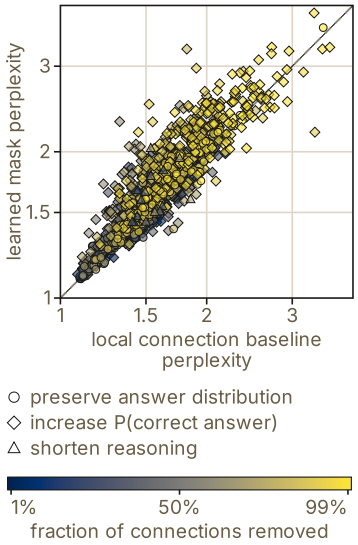
Fixed-suffix evaluation. After training, we fix M and sample a test set $\mathcal{D}_{\text{test}}$ of suffixes independently from $p_{\text{clean}}(Y \mid x)$. We supply each recorded suffix to the masked model and compute

$$\mathcal{L}_{\text{fixed}}(M; x) = \underset{y\in\mathcal{D}_{\text{test}}}{\text{median}}\ \ell(M; x, y). \quad (4)$$

Every mask is evaluated on the same test suffixes. This measures how well the circuit satisfies the objective on additional continuations from the unmasked model.

	fixed suffixes	generated suffixes
	normalized KL ↓ ($\times 10^{-3}$)	$\mathrm{KL}(\bar{q}_{\mathrm{clean}} \,\|\, \bar{q}_M)$ ↓ ($\times 10^{-3}$)
Preserving the answer distribution		
learned mask	**7.6 ±2.5**	**2.1 ±1.4**
local connection baseline	16.0 ±5.4	3.5 ±1.8
random mask	79.9 ±24.4	8.3 ±7.9
unmasked model	–	0.8 ±1.2
	training suffix	generated suffixes
	P(correct answer) ↑	P(correct answer) ↑
Increasing correct-answer probability		
learned mask	**0.68 ±0.10**	**0.54 ±0.13**
local connection baseline	0.25 ±0.09	0.26 ±0.10
random mask	0.26 ±0.10	0.26 ±0.10
unmasked model	0.21 ±0.09	0.23 ±0.10
	generated suffixes	
	tokens until `</think>` ↓	fraction of correct rollouts ↑
Shortening reasoning		
learned mask	**1,354 ±725**	**0.73 ±0.16**
local connection baseline	3,456 ±551	0.26 ±0.17
random mask	2,898 ±694	0.38 ±0.19
unmasked model	2,837 ±712	0.56 ±0.19

(a) Area under the curve of each task metric.



(b) Perplexity of generated suffixes under the unmasked model, one point per prompt and sparsity.

Figure 3: Task metrics and perplexity under attention masks. (a) Each entry gives the median per-prompt area over sparsity, ±95% bootstrap confidence intervals; Appendix C defines the area calculation. Answer preservation uses 64 held-out fixed suffixes and 32 generated suffixes per model; the correct-answer task uses the training suffix and five generated suffixes. Sampled answer-task suffixes have a 200-token limit. Shortening uses 16 suffixes per mask, ending at `</think>` or 4,096 tokens; the random baseline pools two masks. (b) Perplexity under the clean model for each prompt used in our Generated-suffix evaluations. Rollouts sampled from the local connection baseline (x-axis) are compared with those sampled from the learned mask (y-axis), for all three tasks.

Generated-suffix evaluation. With M fixed, we sample independent suffixes

$$\widetilde{y}_1, \ldots, \widetilde{y}_{N_{\mathrm{gen}}} \sim p_M(Y \mid x) \tag{5}$$

and measure their reasoning outcomes using the task's evaluation metrics. We also generate suffixes from the unmasked model and each baseline mask, using the same number of samples and decoding settings. Both evaluation protocols test generalization to new suffixes from the same prefix. All masks remain fixed during evaluation.

3 Circuit Discovery from Reasoning Outcomes

We run three experiments, each with different desiderata: 1) finding a sparse circuit that preserves the original model's answer distribution, 2) finding a sparse circuit that increases the probability of a correct answer, and 3) finding a sparse circuit that shortens the reasoning chain. We apply the same optimization to find a sparse circuit (Equation 2), fit to a different loss $\ell(\widetilde{M}; x, y)$ per experiment.

Models and data. We study Qwen3-8B and Qwen3-32B (Qwen Team, 2025). For Qwen3-8B, we use GPQA Diamond (Rein et al., 2024), AQuA-RAT (Ling et al., 2017), and MATH (Hendrycks et al., 2021b), whereas for Qwen3-32B, we use GPQA Diamond and MMLU (Hendrycks et al., 2021a). All datasets except MATH have multiple-choice answers. For MATH, we define buckets over unique answers (see Section F.1). We sample 16 unmasked traces for each question at temperature 0.6 and use the first sample for the reasoning prefix x. For the answer tasks, we filter to questions whose empirical accuracy across 16 traces lies between 0.25 and 0.75, in order for circuits to have a signal to optimize for. Additional filters for the length task are given in the following section. Appendix F explains how we sample traces, select prefixes, and filter questions for each task.

Experimental setup. For each prefix and objective, we train a separate mask at each sparsity $s \in \{0.01, 0.05, 0.2, 0.3, \ldots, 0.9, 0.95, 0.99\}$. We compare with the local connection baseline and random masks at the same sparsity. Figure 3a shows the AUCs obtained by these masks on each task and Figure 7 shows the median score per evaluated sparsity.

3.1 Preserving the answer distribution

In this experiment, we optimize for sparse circuits that preserve the answer distribution on continuations sampled from the unmasked model.

We define answer distribution as follows. Let A be the random variable for the answer choice, with values in the set $\mathcal{A}$ of valid answer letters (e.g. $\mathcal{A} = \{\text{A,B,C,D}\}$). After processing the prefix x, suffix y, and a fixed answer prompt, we normalize the model's next-token probabilities over $\mathcal{A}$. We write the resulting distribution as $q_{\widetilde{M}}(A \mid x, y)$, with $q_{\widetilde{M}}(a \mid x, y)$ denoting the probability of a particular answer $a \in \mathcal{A}$. The corresponding distributions under the unmasked model and the trained mask are $q_{\text{clean}}(A \mid x, y)$ and $q_M(A \mid x, y)$.

We sample the 32 suffixes of $\mathcal{D}_{\text{train}}$ and the 64 suffixes of $\mathcal{D}_{\text{test}}$ from $p_{\text{clean}}(Y \mid x)$ and stop each suffix after 200 tokens. We process the same suffix with a candidate mask and minimize

$$\ell_{\text{preserve}}(\widetilde{M}; x, y) = D_{\text{KL}}\big(q_{\text{clean}}(A \mid x, y) \,\big\|\, q_{\widetilde{M}}(A \mid x, y)\big)\,. \tag{6}$$

Each training step draws four suffixes from $\mathcal{D}_{\text{train}}$ without replacement and applies the same four sampled masks to each suffix. For fixed-suffix evaluation, we divide the loss for the trained mask M by the loss obtained when all eligible connections are removed, separately for each test suffix, and report the median over $\mathcal{D}_{\text{test}}$:

$$\text{KL}_{\text{norm}}(M; x) = \underset{y \in \mathcal{D}_{\text{test}}}{\text{median}} \; \frac{\ell_{\text{preserve}}(M; x, y)}{\ell_{\text{preserve}}(\mathbf{0}; x, y)}. \tag{7}$$

Here $\mathbf{0}$ removes all eligible connections in $\mathcal{E}$ (i.e., attention between all reasoning sentence pairs). We drop test suffixes with $\ell_{\text{preserve}}(\mathbf{0}; x, y) = 0$; that is, we ignore cases where dropping all of the connections in $\mathcal{E}$ doesn't affect the answer distribution. A value of zero indicates that the answer distributions are preserved, and a value of one equals the loss of removing every eligible connection.

For the generated-suffix evaluation, we sample 32 suffixes $\widetilde{y}_1, \ldots, \widetilde{y}_{32} \sim p_M(Y \mid x)$ of 200 tokens and 32 reference suffixes $y'_1, \ldots, y'_{32} \sim p_{\text{clean}}(Y \mid x)$, disjoint from $\mathcal{D}_{\text{train}}$, and compare the mean answer distributions

$$D_{\text{KL}}(\bar{q}_{\text{clean}} \,\|\, \bar{q}_M)\,, \quad \bar{q}_{\text{clean}} = \tfrac{1}{32}\textstyle\sum_k q_{\text{clean}}(A \mid x, y'_k), \quad \bar{q}_M = \tfrac{1}{32}\textstyle\sum_k q_M(A \mid x, \widetilde{y}_k). \tag{8}$$

For the unmasked model, $\widetilde{y}_k$ are 32 further unmasked suffixes, separate from the held out D_{test} from fixed suffix evaluation. So we can use it to measure sampling noise in Figure 3a.

Results. At 20–90% sparsity, the local connection baseline has 1.7–3.6 times the median normalized KL of the learned masks on held-out fixed suffixes (Figure 7a, left). On generated suffixes, the baseline has 1.2–3.2 times the median KL between mean answer distributions (Figure 7a, right).

3.2 Increasing correct-answer probability

In this experiment, we learn a sparse attention circuit that increases the probability of generating a correct answer at the end of the reasoning chain.

Let $a^\star \in \mathcal{A}$ be the correct answer. We use the answer probabilities defined above and train the mask to increase the probability of $a^\star$ relative to the strongest incorrect answer:

$$\ell_{\text{correct}}(\widetilde{M}; x, y) = \max_{a \in \mathcal{A} \setminus \{a^\star\}} q_{\widetilde{M}}(a \mid x, y) - q_{\widetilde{M}}(a^\star \mid x, y). \tag{9}$$

This task trains on a single suffix, $\mathcal{D}_{\text{train}} = \{y_{\text{orig}}\}$, where y_{orig} holds the five sentences that follow x in the original trace, and samples one mask per training step. Figure 7b shows the training performance of this task, $q_M(a^\star \mid x, y_{\text{orig}})$ together with the unmasked value $q_{\text{clean}}(a^\star \mid x, y_{\text{orig}})$ (left), and its generated-suffix evaluation (right). Generated-suffix evaluation samples five suffixes

$\widetilde{y}_1, \ldots, \widetilde{y}_5 \sim p_M(Y \mid x)$ of 200 tokens, appends the fixed answer prompt to each, and reports the median correct-answer probability under the same mask:

$$q_M^{\text{gen}}(a^\star \mid x) = \underset{k=1,\ldots,5}{\text{median}}\ q_M(a^\star \mid x, \widetilde{y}_k). \tag{10}$$

The unmasked reference replaces M with the unmasked model in both sampling and scoring and uses the same random seeds. Baseline masks replace M in the same way.

Results. Correct-answer probability on generated suffixes peaks at around 0.72–0.77 at 40–60% against 0.21–0.24 for the local baseline and 0.23 for the unmasked model (Figure 7b, right).

3.3 Shortening reasoning

We train a mask to reduce continuation length while also increasing the probability of a correct answer. For this task, we use questions with empirical accuracy between $[0.5, 0.75]$ across reasoning traces, and use a correct original trace for our prefix, such that unmasked continuations from the prefix have variability in their final answers. We sample 12 training continuations from the unmasked model at temperature 1.0, each stopping after an end-of-thinking (`</think>`) token or after $H = 4{,}096$ tokens have been generated.

For each training suffix y, we compute the masked model's probability of terminating reasoning after every paragraph break. A break is eligible only when the unmasked model, prompted to answer there, puts the most probability on the correct answer – which is also, by design, the same answer as the original reasoning trace. These probabilities define a stopping model over eligible breaks. Let $T_{\widetilde{M}}(x, y)$ be the random stopping position, measured in tokens from the start of the suffix, with value $T_{\widetilde{M}}(x, y) = H$ if the reasoning never reaches termination. The length term is

$$\ell_{\text{length}}(\widetilde{M}; x, y) = \frac{1}{H}\mathbb{E}\left[T_{\widetilde{M}}(x, y)\right]. \tag{11}$$

Appendix B.4 defines the closing tokens for reasoning and gives the expectation in closed form. We exclude suffixes without an eligible break from the training average.

The task loss is ℓ_{length}. Stopping counts only at eligible breaks, so minimizing ℓ_{length} selects early stops at which the correct answer is the most probable answer. Figure 7c shows the median length and the fraction of correct answers over 16 newly generated rollouts per mask.

For generated-suffix evaluations we report the median number of tokens until `</think>`, assigning H to continuations that reach the horizon, and the fraction of rollouts that output the correct final answer. Again, for chains that do not end under our token-budget, we score it as 'incorrect'.

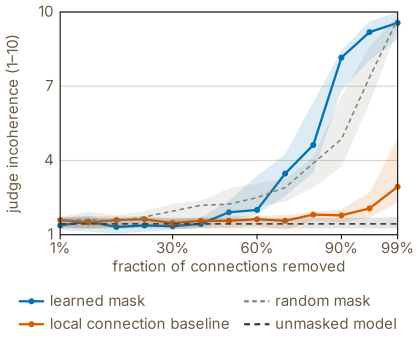


Figure 4: Judge incoherence ratings (Gemini 3.8 Flash) of the reasoning generated under each mask on the reasoning brevity objective (Section 3.3). Our learned mask finds important connections for preserving coherence. Removing 90% of those connections hurt coherence worse than removing a random 90% of the connections; meanwhile, removing 90% of the connections found by the local baseline doesn't significantly reduce coherence. On other two tasks, whose suffixes stop after 200 tokens, the judge rates every mask coherent at all sparsities (median ≤ 1.25).

Results. Generated continuation length decreases from approximately 2,800 to fewer than 1,000 tokens at 30–80% sparsity while maintaining the original answer (Figure 7c). Local and random masks increase continuation length toward the horizon. However, we also observe that with larger sparsities, rollouts optimized for shortening reasoning become increasingly incoherent (Fig. 4). This is not the case for our other two optimization objectives, where rollouts remain coherent across until sparsity levels of $\geq 99\%$. We can also see in Figure 7c that this is related to the general failure of masks on the task metric at these sparsities.

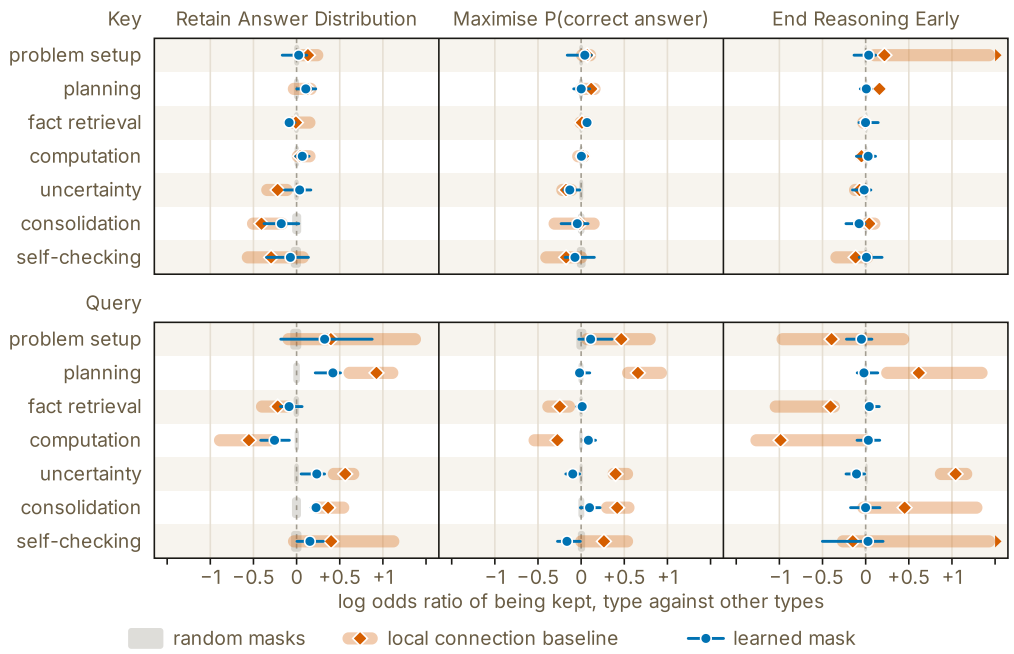


Figure 5: Sentence types of the connections that each mask keeps. Each connection lets the tokens of a later reasoning sentence (Query) attend to an earlier reasoning sentence (Key). For each sentence type from Bogdan et al. (2025), we report the log odds ratio that a mask keeps a connection whose Key (or Query) has that type versus a different type. A value of zero means that the mask keeps connections of that sentence type equally as often as for other sentence types. Whereas the local connection method keeps *planning* and *uncertainty* sentence categories more often, our learned mask doesn't consistently prefer a specific sentence type across different objectives.

4 Investigating Learned Attention Circuits

Next, we discuss qualitative findings which we observed across the circuits we identified in these three domains. These findings not only demonstrate the general utility of our method, but they also shed light on the fundamental nature of reasoning model behavior in general.

The attention circuit is distributed across reasoning steps with no clear important steps. Analyzing the attention patterns of the circuits identified by our method, we find that attention is distributed across steps throughout a reasoning chain (Fig. 1c). Previous work on attention circuits in reasoning found that local connections in reasoning steps were most effective in maintaining model performance (Bogdan et al., 2025), which would suggest that each reasoning step builds directly on the preceding step in a sequential manner. By contrast, our results suggest that important steps in a reasoning chain are scattered throughout the chain, with no clear pattern. Further, when we tasked an external LLM judge with identifying the most important steps in each reasoning chain (Fig. 15), there was little correlation with the attention circuits identified by our learned mask method, except in the Retain Answer Distribution experiment.

Learned attention circuits are unstable across random seeds. Different training seeds find different attention masks, often with similar performance at the same sparsity level (Fig. 6). This suggests that reasoning chains have lots of redundant information that allows for many different circuits to behave similarly.

Attention circuits we identify are distributed uniformly across categories of reasoning steps. We used an LLM judge to label every individual step in each reasoning chain according to seven categories: *problem setup, planning, fact retrieval, computation, uncertainty management, consolidation*, and *self-checking*, following the grading rubric of Bogdan et al. (2025). Whereas Bogdan

et al. (2025) found that their attention masking disproportionately preserved attention to *planning* and *uncertainty management* sentences (and our local connection baseline replicates this result), our learned mask method produces a very different pattern, with nearly uniform attention to and from all categories of reasoning steps. See Fig. 5 (local connection baseline in orange, our learned mask in blue) for a breakdown of preserved attention connections by sentence category.

5 Discussion

Attention circuits learned via desiderata-based masking outperform baselines. We presented experimental results where we applied this method to three different loss functions, which optimize for maintaining the distribution over final answers, for increasing the probability of the correct answer, and for reasoning brevity. We evaluated our method on both fixed reasoning chains, generated without masking, as well as reasoning chains generated with masking. In all cases, our model significantly outperforms the attention masking baseline developed in prior work, as well as a random masking baseline.

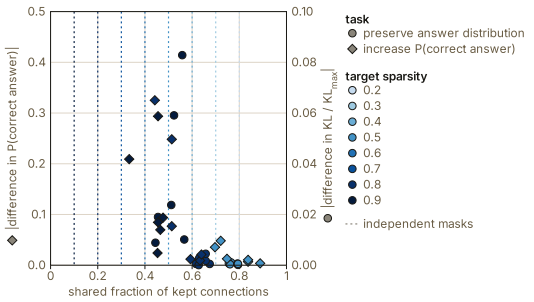


Figure 6: Agreement between masks trained across three seeds (Qwen3-8B, one GPQA Diamond and one AQuA-RAT prompt). Each point compares one pair of seeds at one prompt and target sparsity. The x-axis gives the fraction of the connections kept by one mask that the other mask also keeps. The left axis gives the absolute difference in correct-answer probability on the training suffix (diamonds); the right axis gives the absolute difference in normalized KL on the 32 training rollouts (circles). Dotted lines mark the expected shared fraction $1 - s$ of two random masks at each target sparsity s.

Attention circuits suggest that reasoning is messy, distributed, and redundant. Our method discovers attention circuits that are distributed throughout a reasoning chain and have long-range dependencies, including reasoning steps of various categories, and that many redundant circuits exist within a reasoning chain. These results go directly against the claims of (Bogdan et al., 2025) that reasoning is punctuated by important "thought anchor" steps which carry key information in a structured and predictable manner. We speculate that this is a way that reasoning models truly diverge from traditional LLMs, since their reinforcement learning post-training has no incentive to produce structured, coherent lines of reasoning more akin to patterns in their (human-generated) pre-training data. Instead, reasoning models end up producing a thick soup of plans, facts, self-checking, 'aha' moments, and so on in their hidden reasoning.

Attention circuits might have future applications in monitoring, alignment, or post-training. We see a number of exciting future directions for our work. First, this method may also have useful applications in monitoring reasoning chains through identifying their key parts, for example to verify the faithfulness (Turpin et al., 2023) or bias (Karvonen & Marks, 2025) of a reasoning chain, in ways that are difficult for a judge which only sees the full, unfiltered reasoning text. This may be especially helpful for models where reasoning has devolved into a form that is no longer human interpretable, e.g. as recently evidenced by Panfilov et al. (2026) and Petersson (2026). Next, there may be additional reasoning outcomes which can be used to optimize attention masking, for example optimizing reasoning chains for alignment or faithfulness, which prove useful for understand the circuits that reasoning models use to achieve various complex behaviors. It may even be useful to optimize for harmful objectives, such as hallucinations, to understand how and why reasoning sometimes goes wrong. Finally, this attention masking could be used in post-training objective, so that reasoning models can e.g. generate much shorter chains which nonetheless have the same outcome.

AI use statement

We used LLMs to help implement standard metrics, optimize methods for GPU execution, and run hyperparameter sweeps. We also used LLMs to draft appendix sections and check that the

experimental details matched the implementation. We take responsibility for the final content of the paper and the accompanying code.

Reproducibility statement

We provide code in an anonymous repository. Appendix B describes implementation details, Appendix D reports hyperparameter comparisons, and Appendix F specifies the datasets and prompt selection procedure. Section 3 defines the task objectives and evaluation protocols. We also release a dashboard containing generated rollouts here: https://reasoning-attention-circuits.netlify.app/

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, et al. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1z-PsR5KX.

Eric Bigelow, Ari Holtzman, Hidenori Tanaka, and Tomer Ullman. Forking paths in neural text generation. *arXiv preprint arXiv:2412.07961*, 2024.

Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which LLM reasoning steps matter? *arXiv preprint arXiv:2506.19143*, 2025.

Siddharth Boppana, Annabel Ma, Max Loeffler, Raphael Sarfati, Eric Bigelow, Atticus Geiger, Owen Lewis, and Jack Merullo. Reasoning theater: Disentangling model beliefs from chain-of-thought. *arXiv preprint arXiv:2603.05488*, 2026.

Steven Cao, Victor Sanh, and Alexander M. Rush. Low-complexity probing via finding subnetworks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 960–966, 2021.

Arthur Conmy, Augustine N. Mavor-Parker, Aidan Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. ACDC: Automatic circuit DisCovery. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Xander Davies, Max Nadeau, Nikhil Prakash, Tamar Rott Shaham, and David Bau. Discovering variable binding circuitry with desiderata. *arXiv preprint arXiv:2307.03637*, 2023. URL https://arxiv.org/abs/2307.03637.

Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. How do decisions emerge across layers in neural models? interpretation with differentiable masking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3243–3255. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.262. URL https://aclanthology.org/2020.emnlp-main.262.

Nicola De Cao, Leon Schmid, Dieuwke Hupkes, and Ivan Titov. Sparse interventions in language models with differentiable masking. *arXiv preprint arXiv:2112.06837*, 2021. URL https://arxiv.org/abs/2112.06837.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175, 2021. doi: 10.1162/tacl_a_00359. URL https://aclanthology.org/2021.tacl-1.10/.

Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart Shieber, Tal Linzen, and Yonatan Belinkov. Causal analysis of syntactic agreement mechanisms in neural language models. In *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. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.144. URL https://aclanthology.org/2021.acl-long.144/.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah D. Goodman. Finding alignments between interpretable causal variables and distributed neural representations. In *Proceedings of the Third Conference on Causal Learning and Reasoning*, volume 236, pp. 160–187. PMLR, 2024. URL https://proceedings.mlr.press/v236/geiger24a.html.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64, 2025. URL https://jmlr.org/papers/v26/23-0058.html.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021b.

Jing Huang, Zhengxuan Wu, Christopher Potts, Mor Geva, and Atticus Geiger. RAVEL: Evaluating interpretability methods on disentangling language model representations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8669–8687. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.470. URL https://aclanthology.org/2024.acl-long.470/.

Adam Karvonen and Samuel Marks. Robustly improving LLM fairness in realistic settings via interpretability. *arXiv preprint arXiv:2506.10922*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018.

Uzay Macar, Paul C. Bogdan, Senthooran Rajamanoharan, and Neel Nanda. Thought branches: Interpreting LLM reasoning requires resampling. *arXiv preprint arXiv:2510.27484*, 2025.

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. *arXiv preprint arXiv:2403.19647*, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Alexander Panfilov, David Schmotz, Ilia Shumailov, Luca Beurer-Kellner, Joachim Schaeffer, Ameya Prabhu, Jonas Geiping, and Maksym Andriushchenko. Stealing reasoning traces from proprietary llm apis. *arXiv preprint arXiv:2608.09867*, 2026.

Lukas Petersson. Cot monitoring is dying. gpt-6 astra communicates with its sub-agents in text that is barely understandable for humans. X (formerly Twitter), 2026. URL https://x.com/johnschulman2/status/2097494108491403268. Accessed: 2026-09-25.

Nikhil Prakash, Natalie Shapira, Arnab Sen Sharma, Christoph Riedl, Yonatan Belinkov, Tamar Rott Shaham, David Bau, and Atticus Geiger. Language models use lookbacks to track beliefs, 2025. URL https://arxiv.org/abs/2505.14685.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3319–3328, 2017.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. *arXiv preprint arXiv:2310.10348*, 2023.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, 2023.

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 *Advances in Neural Information Processing Systems*, volume 33, pp. 12388–12401, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/92650b2e92217715fe312e6fa7b90d82-Abstract.html.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023.

Zhengxuan Wu, Atticus Geiger, Thomas Icard, Christopher Potts, and Noah D. Goodman. Interpretability at scale: Identifying causal mechanisms in Alpaca. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f6a8b109d4d4fd64c75e94aaf85d9697-Abstract-Conference.html.

Jiachen Zhao, Yiyou Sun, Weiyan Shi, and Dawn Song. Can aha moments be fake? identifying true and decorative thinking steps in chain-of-thought. *arXiv preprint arXiv:2510.24941*, 2025.

Amir Zur, Atticus Geiger, Ekdeep Singh Lubana, and Eric Bigelow. Are language models aware of the road not taken? token-level uncertainty and hidden state dynamics. *arXiv preprint arXiv:2511.04527*, 2025.

A Related Work

Causal Analysis and Circuits. Causal analysis evaluates hypotheses about model computation by testing the effects of interventions on internal representations (Bau et al., 2019; Elazar et al., 2021; Finlayson et al., 2021; Geiger et al., 2025; Meng et al., 2022; Vig et al., 2020). Circuit discovery identifies subgraphs of neural network computation graphs that mediate specified model behaviors (Ameisen et al., 2025; Conmy et al., 2023; Marks et al., 2024; Syed et al., 2023; Wang et al., 2023).

Learning interventions from behavioral objectives. Desiderata-based masking learns differentiable masks over model components by optimizing objectives on their intervention effects (Cao et al., 2021; Davies et al., 2023; De Cao et al., 2020; 2021; Prakash et al., 2025). Distributed alignment search learns subspaces whose interchange-intervention effects match those specified by a high-level causal model (Geiger et al., 2024; Huang et al., 2024; Wu et al., 2023).

Interpretability of reasoning models. Sentence-level interventions and resampling measure dependencies within reasoning traces and effects on final-answer distributions. Hidden-state analyses and reasoning-step perturbations track answer information, predict future outcomes, and support early stopping or pruning (Bigelow et al., 2024; Bogdan et al., 2025; Boppana et al., 2026; Macar et al., 2025; Zhao et al., 2025; Zur et al., 2025).

B METHOD DETAILS

B.1 SENTENCE SPLITTING

A sentence is a span of tokens that ends at a full stop, question mark, exclamation mark, newline, or a `<think>`/`</think>` token. We merge spans shorter than 10 tokens with the next span. Suffix tokens retain causal attention to the prefix and earlier suffix tokens in both evaluation protocols.

B.2 LOCAL CONNECTION BASELINE

The local connection baseline uses one unmasked forward pass and one ablated pass per prefix sentence S_j. Each ablated pass removes attention from prefix query tokens to S_j in every layer and head. We compute the next-token KL divergence over the full vocabulary at each position, then average over each later sentence S_i to obtain c_{ij} (Equation (3)). The baseline retains the k highest-scoring connections in $\mathcal{E}$; one score matrix serves every objective and sparsity for a given prefix. Unlike Bogdan et al. (2025), we average KL divergences without taking their logarithms or subtracting a mean for each query sentence.

B.3 SUBNETWORK PROBING WITH HARD-CONCRETE MASK VALUES

Each eligible connection $(i, j) \in \mathcal{E}$ has a learnable scalar $\phi_{ij} = \log \alpha_{ij} \in \mathbb{R}$, the location of its hard-concrete distribution (Louizos et al., 2018). We sample candidate mask values as follows:

$$u_{ij} \sim \mathcal{U}(\epsilon, 1-\epsilon), \qquad v_{ij} = \sigma\left(\frac{\log u_{ij} - \log(1-u_{ij}) + \phi_{ij}}{\beta}\right), \tag{12}$$

$$\widetilde{M}_{ij} = \mathrm{clip}_{[0,1]}\big(v_{ij}\,(\zeta - \gamma) + \gamma\big), \tag{13}$$

with $\beta = 2/3$, $\gamma = -0.1$, $\zeta = 1.1$ and $\epsilon = 10^{-6}$. The stretch to $[\gamma, \zeta]$ followed by clipping gives the mask value a non-zero probability of being exactly 0 or exactly 1. The activation probability is $\pi_{ij}(\phi) = \sigma(\phi_{ij} - \beta \log(-\gamma/\zeta))$, up to the truncation of u_{ij} by ϵ. Ranking connections by ϕ_{ij} gives the same order as ranking by $\pi_{ij}(\phi)$ for the top-k selection in Section 2. For the correct-answer and shortening tasks, we initialize every ϕ_{ij} at 2. The answer-preservation masks shown in Figure 7a start from the local connection baseline: $\phi_{ij} = 2$ for its top k connections and -2 for the rest. Appendix D.5 compares the two initializations. The sampled mask value enters the attention logits as $\log \widetilde{M}_{ij}$, with $\log 0$ floored at $\log 10^{-30}$ (Equation (1)). Each answer-preservation step draws four training suffixes without replacement and applies the same four sampled masks to each suffix. The step averages the 16 task losses. Each correct-answer step applies one sampled mask to the stored training suffix; each shortening step applies one sampled mask to all eligible training suffixes.

Sparsity penalty. Cao et al. (2021) use $\frac{1}{|\mathcal{E}|}\sum_{(i,j)\in\mathcal{E}} \pi_{ij}(\phi)$, the mean probability that a connection remains active. The penalty favors fewer active connections at every sparsity, so reaching a target budget requires tuning λ. The quadratic penalty in Equation (2) instead penalizes the squared distance between the expected number of active connections and k. The quadratic gradient pushes the expected count toward k from either side.

Sparsity weight and optimizer. The weight λ of Equation (2) is 0 for the first 25% of the training steps, rises linearly to $\lambda_{\max}$ over the next 50%, and stays at $\lambda_{\max}$ for the rest. All three tasks use $\lambda_{\max} = 1000$ with the quadratic (L2) sparsity penalty. The answer tasks use 1000 training steps, and shortening uses 500. Each step updates ϕ with Adam on the task loss and plain gradient descent on the sparsity penalty, both at learning rate 0.1. The separate update preserves the scale of the sparsity gradient without Adam normalization. We freeze model weights, use gradient checkpointing, and split Qwen3-32B across two GPUs.

B.4 SHORTENING OBJECTIVE DETAILS

Closed form of Equation (11). We process each fixed training suffix y with a candidate mask. At paragraph break b, after t_b suffix tokens, let z_b be the next-token logits over the vocabulary $\mathcal{V}$. The stopping probability uses the unscaled logits:

$$h_b = \sum_{w \in \mathcal{W}} \frac{\exp z_b(w)}{\sum_{v \in \mathcal{V}} \exp z_b(v)}, \tag{14}$$

where $\mathcal{W}$ is the set of tokens that start the closing sequence. The eligible set $\mathcal{B}^+$ contains breaks where the unmasked model assigns the highest answer probability to $a^\star$ after the fixed answer prompt. We interpret h_b as the conditional probability of stopping at break b after passing all earlier eligible breaks. The expected length is

$$\ell_{\text{length}}(\widetilde{M}; x, y) = \frac{1}{H}\Big(\sum_{b \in \mathcal{B}^+} R_b \, h_b \, t_b \; + \; R_{\text{end}} \, H \Big), \qquad R_b = \prod_{b' \in \mathcal{B}^+,\, b' < b} (1 - h_{b'}), \tag{15}$$

where R_b is the probability of reaching break b and $R_{\text{end}} = \prod_{b \in \mathcal{B}^+}(1 - h_b)$ is the probability of passing every eligible break. The final term assigns the horizon $H = 4{,}096$ to the remaining probability mass. The task loss averages Equation (15) over suffixes with eligible breaks.

Closing tokens and evaluation. The closing set $\mathcal{W}$ contains `</think>` and tokens that begin the final-answer line (usually $\star\star$). The stopping model assigns zero stopping probability to breaks outside $\mathcal{B}^+$; generated suffixes have no such constraint. Generated-suffix evaluation samples 16 continuations per mask at temperature 1.0 and stops at `</think>` or H tokens. We report the median length and the fraction of all continuations that stop and give the correct answer. We read the answer after appending the fixed answer prompt to each stopped suffix. Continuations that reach H count as length H and as incorrect. The random baseline pools 16 continuations from each of two masks, for 32 continuations per prompt and sparsity. The shortening sweep also includes $s = 0.1$.

C TASK METRICS ACROSS SPARSITIES

Figure 7 shows the task metrics at each target sparsity. For each prompt, we summarize suffixes using the task-specific rule in Section 3, then report the median across prompts. Answer preservation uses held-out fixed suffixes and newly generated suffixes. The correct-answer task reports the training suffix and newly generated suffixes; shortening reports only newly generated suffixes.

The areas in Figure 3a summarize each prompt separately before aggregation across prompts. We integrate each metric over sparsity with the trapezoidal rule and divide by the width of the sparsity grid. For KL, we floor values at 10^{-6}, integrate $\log_{10}$ KL, and report 10^{area}. The table gives the median across prompts, with an error term equal to half the width of the 95% bootstrap interval. An area computed from the median curve in Figure 7 need not equal the median of the per-prompt areas.

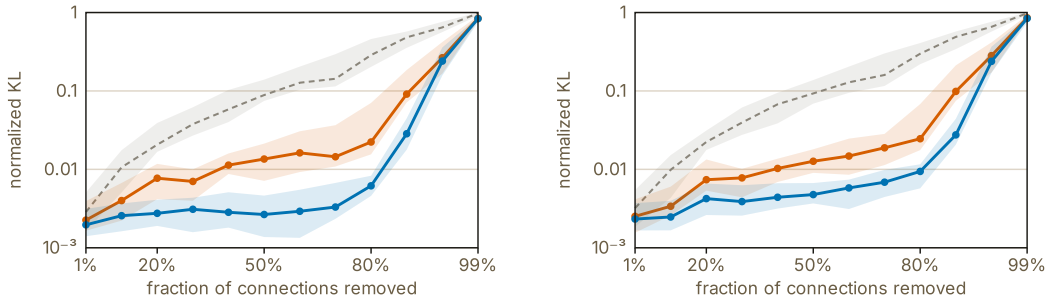
D HYPERPARAMETER TUNING AND ALTERNATIVE TRAINING METHODS

We compare alternative mask optimizers, sparsity penalties, and training-set sizes. All methods use sentence-pair masks and retain the top k connections at evaluation. The studies below use separate prompt subsets and evaluation protocols, which we specify for each comparison.

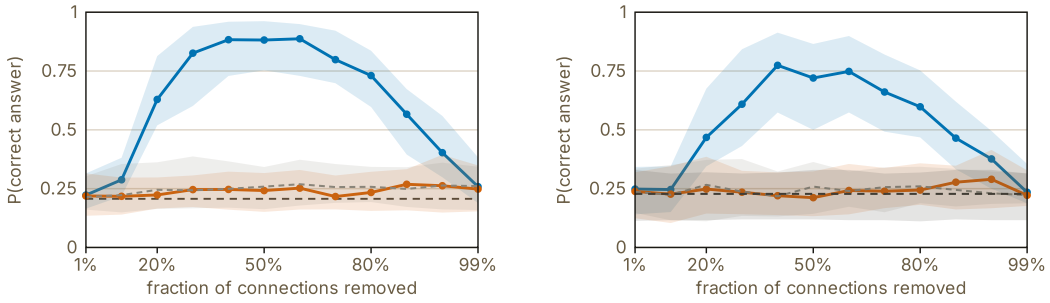
D.1 ALTERNATIVE METHODS

Deterministic continuous mask. Each eligible connection has a score θ_{ij}, and the mask value $\widetilde{M}_{ij} = \sigma(\theta_{ij})$ enters Equation (1) directly, with no sampling. The penalty applies the quadratic budget of Equation (2) to the mask values and adds a term that pushes every value toward 0 or 1:

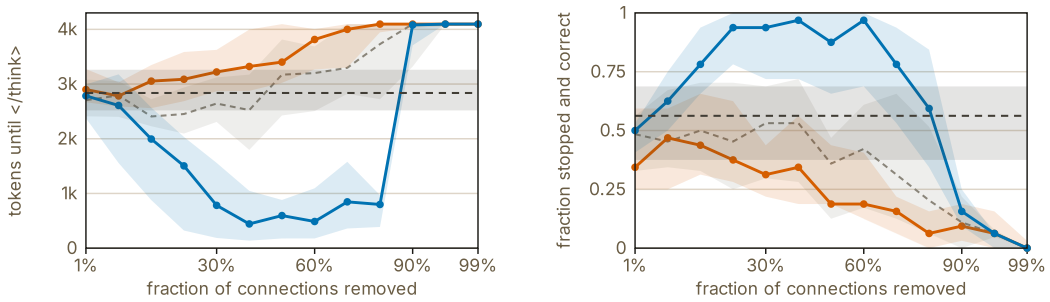
$$\lambda(t) \left[\frac{1}{|\mathcal{E}|}\Big(\sum_{(i,j) \in \mathcal{E}} \widetilde{M}_{ij} - k \Big)^2 + \frac{c}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} \widetilde{M}_{ij}\big(1 - \widetilde{M}_{ij}\big) \right], \tag{16}$$



(a) Answer preservation. Left: normalized KL on 64 held-out fixed suffixes (Equation (7)). Right: KL between mean answer distributions from 32 unmasked and 32 masked-model generated suffixes (Equation (8)).



(b) Correct-answer probability. Left: the training suffix. Right: the median over five generated suffixes (Equation (10)).



(c) Shortening reasoning. Left: median tokens until `</think>`, with $H = 4{,}096$ for suffixes that reach the limit. Right: fraction of all suffixes that stop and give the correct answer.

Figure 7: Task metrics per sparsity for the AUCs in Figure 3a. Curves show medians across prompts, with 95% bootstrap confidence intervals.

with $c = 1$, $\lambda_{\max} = 1000$ and $\lambda(t)$ rising linearly from zero over the first half of training. We trained the continuous mask on 32 suffixes for eight prompt–sparsity pairs. On a separate set of 32 held-out suffixes, the continuous-mask method had 1.2–7.5 times the normalized KL of hard-concrete sampling (median ratio 3.6). Binary evaluation replaces the continuous training mask.

Deterministic continuous mask with a PID controller (DCM + PID). Each eligible connection has a mask value $m_{ij} \in [0, 1]$, initialized at 0.99 and used directly as $\widetilde{M}_{ij}$. Adam updates m with the gradient of the task loss. A separate gradient step of size $\eta\mu/|\mathcal{E}|$ lowers every mask value, where η is the learning rate and μ a multiplier. A proportional-integral-derivative (PID) controller on $\log \mu$

sets μ so that the fraction of connections with $m_{ij} < 0.5$ follows a linear ramp from 0 to 0.95 over training. A second step of size $\eta c\,(2m_{ij} - 1)$, with $c = 0.1$, moves each value toward the nearer of 0 and 1 (*polarized* in Figure 11). Each time the fraction of connections below 0.5 crosses a target sparsity, the controller holds the ramp for 50 steps (*hold*) and then saves the current mask. One run of 1000 steps therefore yields masks at every target sparsity. The variants in Figure 11 add polarization, the hold, a learning rate of 0.3 instead of 0.1, or a learning rate that rises from 0.05 to 0.3 over the first 30% of training (*rate warmup*).

Attribution patching. Attribution patching (Syed et al., 2023) scores each eligible connection with integrated gradients (Sundararajan et al., 2017) along the path that scales every eligible connection from removed to intact. Let $\alpha\mathbf{1}$ denote the candidate mask with $\widetilde{M}_{ij} = \alpha$ for every $(i, j) \in \mathcal{E}$. The score of connection (i, j) is

$$a_{ij} = -\frac{1}{N+1} \sum_{\alpha \in \{\epsilon,\, 1/N,\, 2/N,\, \ldots,\, 1\}} \frac{\partial\, \ell(\alpha\mathbf{1}; x, y)}{\partial \widetilde{M}_{ij}}, \tag{17}$$

with $N = 10$ steps and $\epsilon = 10^{-10}$ in place of the fully removed endpoint, where $\log 0$ has no gradient. One value $\widetilde{M}_{ij}$ controls every query–key token pair in $S_i \times S_j$, so the derivative sums their contributions. The mask keeps the k connections with the largest a_{ij}. Scoring costs $N + 1$ forward and backward passes per training suffix, and one score matrix serves every sparsity.

Straight-through top-k. Each eligible connection has a score θ_{ij}, initialized at 0 plus Gaussian noise with standard deviation 0.01. The forward pass assigns value 1 to the k_t highest-scoring connections and 10^{-4} to the rest. The number removed, $|\mathcal{E}| - k_t$, rises linearly from zero at the first step to $|\mathcal{E}| - k$ at the middle of training and stays there. The nonzero value for removed connections keeps the gradient of $\log \widetilde{M}_{ij}$ finite during training. The backward pass copies the gradient of the loss with respect to each mask value, kept or removed, to its score. Each step selects exactly k_t connections, so the trainer needs no sparsity penalty.

D.2 Choosing the sparsity penalty

We compared two penalties on the expected number of active connections $n(\phi) = \sum_{(i,j)\in\mathcal{E}} \pi_{ij}(\phi)$. The hinge $\lambda \max(0, n(\phi) - k)$ pushes only while the mask keeps more than k connections. The quadratic penalty is the one in Equation (2). Figure 8 plots the achieved sparsity at the last training step against the target for 8,052 subnetwork-probing runs, pooled over all tuning runs, models and datasets. We measure the achieved sparsity as the fraction of eligible connections whose noise-free mask value, $\mathrm{clip}_{[0,1]}(\sigma(\phi_{ij}/\beta)(\zeta - \gamma) + \gamma)$, equals zero.

The hinge misses the target in both directions. Below the budget the hinge exerts no force, so the task loss alone sets how many connections a run removes. At $\lambda = 100$, runs of the answer-preservation task with targets 0.01 and 0.05 remove a median 0.15 of the connections, and correct-answer runs remove 0.48 to 0.50 for every target up to 0.4. Above the budget the hinge pushes with a constant force, so runs stall short of high targets (median 0.95 at target 0.99). The top-k readout must then change the set of active connections to meet the budget. The quadratic penalty pulls toward the budget from both sides. At $\lambda = 1000$, 74% of quadratic-penalty runs end within 0.02 of the target on each task, with a median absolute deviation of 0.012 for answer preservation and 0.010 for the correct-answer task. Only targets 0.95 and 0.99 fall short, at medians of 0.93 and 0.95 to 0.96. Raising λ to 10^4 brings these runs to 0.94 and 0.97, and $3 \cdot 10^4$ to 0.95 and 0.98. All three tasks use the quadratic penalty with $\lambda_{\max} = 1000$.

D.3 Comparing the training methods

We compared 19 training configurations, the local connection baseline and random masks on the answer-preservation task in a pilot study with six prompts: two GPQA Diamond and two AQuA-RAT prompts for Qwen3-8B and two GPQA Diamond prompts for Qwen3-32B. We cut each prompt after sentence 50 and compare methods at target sparsities 0.1, 0.25, 0.5, 0.75, and 0.9. Three settings differ from Section 3.1. The eligible set includes causal connections involving prompt sentences. Every mask trains on the single stored suffix y_{orig} ($|\mathcal{D}_{\mathrm{train}}| = 1$), and we score normalized KL on

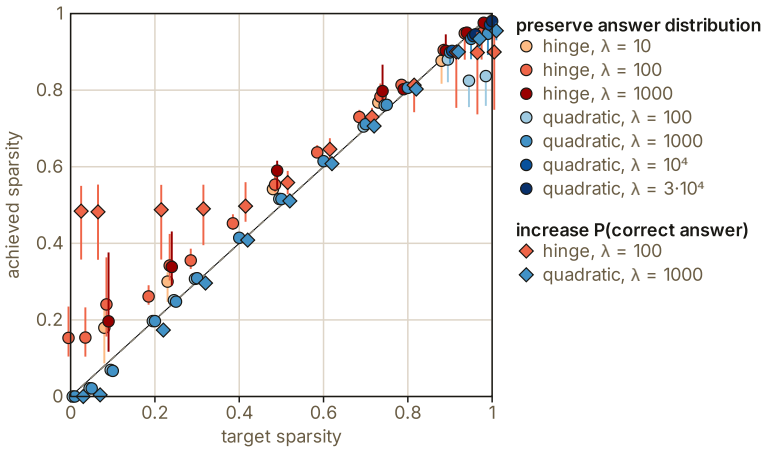


Figure 8: Achieved sparsity at the end of training against the target for subnetwork-probing runs on the answer-preservation task (circles, 5,136 runs) and the correct-answer task (diamonds, 2,916 runs), with the hinge (orange) and quadratic (blue) penalties; darker colours mark larger λ. Markers show the median over runs at each target and whiskers the 10th to 90th percentiles. Horizontal offsets separate the groups.

the same suffix. Subnetwork-probing runs take 1000 steps with learning rate 0.1. In the *canonical* schedule, λ is zero for the first 25% of steps and rises linearly to $\lambda_{\max}$ at 75%.

For methods A and B, let $L_A(c)$ and $L_B(c)$ be their normalized KL for prompt–sparsity pair c, floored at 10^{-6}. Each matrix entry in Figures 9 to 12 reports the median of $\log_{10}\left(L_A(c)/L_B(c)\right)$ over the 30 pairs, with A the row method and B the column method. Each entry also gives an uncorrected two-sided Wilcoxon signed-rank p-value on the paired log differences. A negative median (red) means that the row method has the lower KL, and a border marks $p < 0.05$. The last two columns count the other methods in the same matrix that the row method beats or loses to at $p < 0.05$.

Hard-concrete samples per step (Figure 9). Averaging four or eight masks per step gives lower median KL than one mask per step for each penalty. With the hinge at $\lambda = 100$, four and eight samples beat one sample by median $\log_{10}$ ratios of -0.26 ($p = 0.022$) and -0.22 ($p = 0.013$). With the quadratic penalty, eight samples beat one by -0.34 ($p = 0.001$) at $\lambda = 100$ and by -0.53 ($p = 0.004$) at $\lambda = 1000$. No comparison of eight against four masks reaches $p < 0.05$. We use four masks per step for answer preservation, which requires half as many mask evaluations as eight masks.

Sparsity weight and penalty (Figure 10). Raising λ from 100 to 1000 changes the KL by at most 0.03 in median $\log_{10}$ ratio for every penalty and sample count ($p \geq 0.57$). Against the hinge at equal λ and sample count, the quadratic penalty has median $\log_{10}$ ratios between -0.25 and 0.00, and none reaches $p < 0.05$. We choose the quadratic penalty because its achieved sparsity follows the target more closely than the hinge (Section D.2); the pilot detects no KL difference between the penalties.

DCM + PID (Figure 11). Every DCM + PID configuration loses to subnetwork probing with eight samples per step, with the hinge ($p \leq 0.021$) and with the quadratic penalty at $\lambda = 1000$ ($p \leq 0.006$). Polarization lowers the KL of DCM + PID (-0.45, $p = 0.045$), and the learning-rate warmup and the hold lower it further ($p \leq 0.025$ against polarization alone).

Other schedule settings (Figure 12a). The pilot detects no KL difference between the canonical schedule and a λ ramp from step 0 to 50% or 90% of training ($p \geq 0.17$). Annealing the hard-concrete temperature from 2 to $2/3$ over the full run gives higher KL than the canonical schedule ($+0.41$ in median $\log_{10}$ ratio, $p = 0.011$). Ending the anneal halfway through training gives $p = 0.08$ against the canonical schedule.

Best configuration of each method (Figure 12b). The quadratic penalty at $\lambda = 1000$ with eight samples has a lower median KL than every other method of the pilot. Its advantage over the hinge with eight samples (-0.17, $p = 0.26$) is not significant, while it beats the best DCM + PID configuration (-0.68, $p = 0.005$), the local connection baseline (-1.37, $p < 0.001$) and random masks (-2.90, $p < 0.001$).

		1	2	3	4	5	6	7	8	9	10	beats	loses
1	canonical λ100	–	+0.14 p=0.700	**+0.26 p=0.022**	**+0.22 p=0.013**	+0.03 p=0.968	**+0.57 p=0.006**	**+0.88 p=0.002**	+0.11 p=0.315	**+0.32 p=0.004**	**+0.56 p<.001**	0	6
2	canonical, 2 samples	-0.14 p=0.700	–	+0.25 p=0.100	**+0.30 p=0.027**	-0.09 p=0.452	**+0.24 p=0.009**	**+0.43 p=0.006**	+0.04 p=0.452	**+0.42 p=0.036**	**+0.45 p<.001**	0	5
3	canonical, 4 samples	**-0.26 p=0.022**	-0.25 p=0.100	–	+0.01 p=0.294	-0.13 p=0.080	+0.19 p=0.430	**+0.38 p=0.043**	-0.28 p=0.145	+0.05 p=0.339	+0.35 p=0.061	1	1
4	canonical, 8 samples	**-0.22 p=0.013**	**-0.30 p=0.027**	-0.01 p=0.294	–	**-0.51 p=0.003**	+0.06 p=0.966	-0.00 p=0.552	**-0.38 p=0.035**	-0.02 p=0.539	+0.17 p=0.262	4	0
5	quadratic λ100	-0.03 p=0.968	+0.09 p=0.452	+0.13 p=0.080	**+0.51 p=0.003**	–	**+0.31 p=0.001**	**+0.34 p<.001**	+0.00 p=0.569	**+0.31 p=0.002**	**+0.65 p<.001**	0	5
6	quadratic λ100, 4 samples	**-0.57 p=0.006**	**-0.24 p=0.009**	-0.19 p=0.430	-0.06 p=0.966	**-0.31 p=0.001**	–	+0.01 p=0.406	-0.22 p=0.084	-0.00 p=0.837	+0.16 p=0.469	3	0
7	quadratic λ100, 8 samples	**-0.88 p=0.002**	**-0.43 p=0.006**	**-0.30 p=0.043**	+0.00 p=0.552	**-0.34 p<.001**	-0.01 p=0.406	–	**-0.65 p=0.011**	+0.04 p=0.469	+0.03 p=0.904	5	0
8	quadratic λ1000	-0.11 p=0.315	-0.04 p=0.452	+0.28 p=0.145	**+0.38 p=0.035**	+0.00 p=0.569	+0.22 p=0.084	**+0.65 p=0.011**	–	+0.15 p=0.058	**+0.53 p=0.004**	0	3
9	quadratic λ1000, 4 samples	**-0.32 p=0.004**	**-0.42 p=0.036**	-0.05 p=0.339	+0.02 p=0.539	**-0.31 p=0.002**	+0.00 p=0.837	-0.04 p=0.469	-0.15 p=0.058	–	+0.22 p=0.280	3	0
10	quadratic λ1000, 8 samples	**-0.56 p<.001**	**-0.45 p<.001**	-0.35 p=0.061	-0.17 p=0.262	**-0.65 p<.001**	-0.16 p=0.469	-0.03 p=0.904	**-0.53 p=0.004**	-0.22 p=0.280	–	4	0

Figure 9: Pairwise comparison of hard-concrete sample counts per step (answer preservation, $|\mathcal{D}_{\text{train}}| = 1$, six prompts, five target sparsities). Entry: median over the 30 cells of $\log_{10}$(row KL / column KL) and the uncorrected Wilcoxon p-value; red means the row method has the lower KL; bordered entries have $p < 0.05$. Column j is the method of row j. *Canonical*: hinge with the canonical λ schedule; one sample per step unless stated.

D.4 Attribution patching on the correct-answer task

Figure 13 compares attribution patching ($N = 10$, $\ell = \ell_{\text{correct}}$) with the learned mask and baselines on GPQA Diamond and AQuA-RAT. The comparison measures correct-answer probability on the training suffix. Across 336 prompt–sparsity pairs at 20–90% sparsity, the median probability is 0.26 for attribution patching, 0.22 for random masks, 0.21 for both the local baseline and unmasked model, and 0.56 for the learned mask. Attribution patching exceeds random masks in 88% of pairs, with a median probability difference of 0.03. The learned mask exceeds attribution patching in 90% of pairs.

D.5 Size of the training set and straight-through top-k

We compare training-set sizes on 19 Qwen3-8B prompts (12 GPQA Diamond, 7 AQuA-RAT) at eight sparsities from 0.2 to 0.9. Masks train on the stored suffix y_{orig}, 32 sampled suffixes, or 128 sampled suffixes. The 128-suffix set contains the 32-suffix set, and both exclude the 64 fixed test suffixes. Training draws four suffixes per step when the training set contains more than one suffix. We compare methods on the same 152 prompt–sparsity pairs using the median normalized KL over the test suffixes.

Training on 32 suffixes gives lower held-out KL than training on the stored suffix in 95% of pairs, by a median factor of 2.9. Expanding the training set from 32 to 128 gives a median KL ratio of 1.01, with lower KL in 50% of pairs. Initializing the 32-suffix masks from the local connection baseline gives lower KL than initializing every ϕ_{ij} at 2 in 74% of pairs. At 70–90% sparsity, the median ratio is 0.47, with lower KL in 98% of pairs. The answer-preservation curves in Figure 7a use 32 training suffixes and baseline initialization.

Figure 14 compares hard-concrete sampling with straight-through top-k on 32 training suffixes. With the default initialization of each optimizer, the median held-out KL ratio (top-k divided by hard-concrete) is 0.97 across all eight sparsities (Wilcoxon $p = 0.7$). Starting both optimizers from the local baseline gives higher KL for top-k in 72% of pairs, with a median ratio of 1.40. The top-k masks

		1	2	3	4	5	6	7	8	beats	loses
1	canonical λ100	–	+0.01 p=0.626	+0.03 p=0.968	+0.11 p=0.315	**+0.57 p=0.006**	**+0.32 p=0.004**	**+0.88 p=0.002**	**+0.56 p<.001**	0	4
2	canonical λ1000	-0.01 p=0.626	–	+0.07 p=0.516	+0.25 p=0.058	**+0.57 p=0.002**	**+0.51 p<.001**	**+1.21 p<.001**	**+0.55 p<.001**	0	4
3	quadratic λ100	-0.03 p=0.968	-0.07 p=0.516	–	+0.00 p=0.569	**+0.31 p=0.001**	**+0.31 p=0.002**	**+0.34 p<.001**	**+0.65 p<.001**	0	4
4	quadratic λ1000	-0.11 p=0.315	-0.25 p=0.058	+0.00 p=0.569	–	+0.22 p=0.084	+0.15 p=0.058	**+0.65 p=0.011**	**+0.53 p=0.004**	0	2
5	quadratic λ100, 4 samples	**-0.57 p=0.006**	**-0.57 p=0.002**	**-0.31 p=0.001**	-0.22 p=0.084	–	-0.00 p=0.837	+0.01 p=0.406	+0.16 p=0.469	3	0
6	quadratic λ1000, 4 samples	**-0.32 p=0.004**	**-0.51 p<.001**	**-0.31 p=0.002**	-0.15 p=0.058	+0.00 p=0.837	–	-0.04 p=0.469	+0.22 p=0.280	3	0
7	quadratic λ100, 8 samples	**-0.88 p=0.002**	**-1.21 p<.001**	**-0.34 p<.001**	**-0.65 p=0.011**	-0.01 p=0.406	+0.04 p=0.469	–	+0.03 p=0.904	4	0
8	quadratic λ1000, 8 samples	**-0.56 p<.001**	**-0.55 p<.001**	**-0.65 p<.001**	**-0.53 p=0.004**	-0.16 p=0.469	-0.22 p=0.280	-0.03 p=0.904	–	4	0

Figure 10: Pairwise comparison of the sparsity weight λ and the penalty at one, four and eight hard-concrete samples per step. Entries as in Figure 9.

		1	2	3	4	5	6	beats	loses
1	DCM+PID plain	–	**+0.45 p=0.045**	**+0.84 p=0.002**	**+0.67 p<.001**	**+1.42 p<.001**	**+1.50 p<.001**	0	5
2	DCM+PID polarized	**-0.45 p=0.045**	–	**+0.52 p=0.025**	**+0.62 p<.001**	**+1.08 p<.001**	**+1.27 p<.001**	1	4
3	DCM+PID + rate warmup	**-0.84 p=0.002**	**-0.52 p=0.025**	–	+0.10 p=0.777	**+0.50 p=0.021**	**+0.54 p=0.006**	2	2
4	DCM+PID + hold, rate 0.3	**-0.67 p<.001**	**-0.62 p<.001**	-0.10 p=0.777	–	**+0.48 p=0.015**	**+0.68 p=0.005**	2	2
5	canonical, 8 samples	**-1.42 p<.001**	**-1.08 p<.001**	**-0.50 p=0.021**	**-0.48 p=0.015**	–	+0.17 p=0.262	4	0
6	quadratic λ1000, 8 samples	**-1.50 p<.001**	**-1.27 p<.001**	**-0.54 p=0.006**	**-0.68 p=0.005**	-0.17 p=0.262	–	4	0

Figure 11: Pairwise comparison of the DCM + PID configurations and subnetwork probing with eight hard-concrete samples per step. *Plain*: no polarization; *hold*: 50-step hold at each target sparsity; *rate warmup*: learning rate rising from 0.05 to 0.3 over the first 30% of training. Entries as in Figure 9.

		1	2	3	4	5	beats	loses
1	canonical λ100	–	-0.17 p=0.247	-0.17 p=0.171	**-0.41 p=0.011**	-0.22 p=0.080	1	0
2	no warmup r50	+0.17 p=0.247	–	-0.02 p=0.922	-0.17 p=0.114	-0.15 p=0.339	0	0
3	no warmup r90	+0.17 p=0.171	+0.02 p=0.922	–	-0.37 p=0.084	-0.11 p=0.465	0	0
4	anneal, full run	**+0.41 p=0.011**	+0.17 p=0.114	+0.37 p=0.084	–	+0.17 p=0.152	0	1
5	anneal to 50%	+0.22 p=0.080	+0.15 p=0.339	+0.11 p=0.465	-0.17 p=0.152	–	0	0

(a) Schedule settings of subnetwork probing with the hinge at $\lambda = 100$. *No warmup r50/r90*: λ rises from step 0 until 50% or 90% of training; *anneal*: hard-concrete temperature falls from 2 to $2/3$ over the full run or by 50% of training (no warmup, r90).

		1	2	3	4	5	beats	loses
1	quadratic λ1000, 8 samples	–	-0.17 p=0.262	**-0.68 p=0.005**	**-1.37 p<.001**	**-2.90 p<.001**	3	0
2	canonical, 8 samples	+0.17 p=0.262	–	**-0.48 p=0.015**	**-0.98 p<.001**	**-2.67 p<.001**	3	0
3	DCM+PID + hold, rate 0.3	**+0.68 p=0.005**	**+0.48 p=0.015**	–	**-0.58 p=0.001**	**-2.12 p<.001**	2	2
4	local connection baseline	**+1.37 p<.001**	**+0.98 p<.001**	**+0.58 p=0.001**	–	**-1.61 p<.001**	1	3
5	random mask	**+2.90 p<.001**	**+2.67 p<.001**	**+2.12 p<.001**	**+1.61 p<.001**	–	0	4

(b) The best configuration of each method, the local connection baseline and random masks.

Figure 12: Pairwise comparisons of the remaining settings. Entries as in Figure 9.

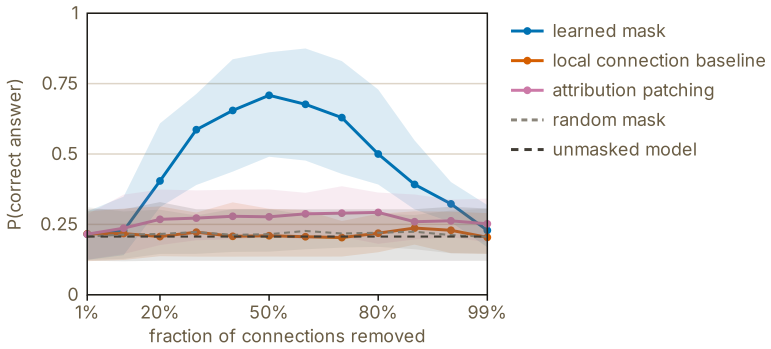


Figure 13: Correct-answer probability on the training suffix over 20 prompts, including attribution patching with 10 integrated-gradient steps.

also retain fewer baseline connections: at 90% sparsity, the median Jaccard index with the baseline mask is 0.14, compared with 0.33 for hard-concrete sampling. The Jaccard index is the number of connections shared by two masks divided by the number kept by either mask. The comparisons support the use of hard-concrete sampling with baseline initialization for answer preservation.

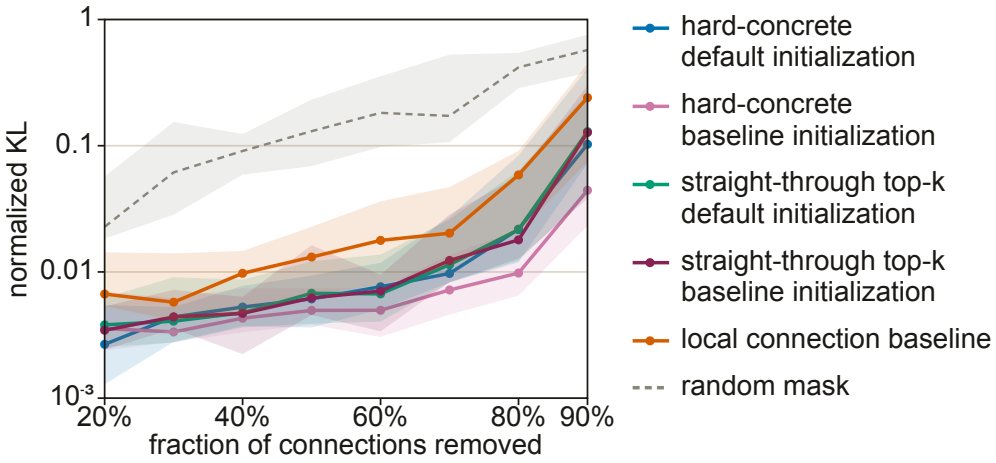


Figure 14: Hard-concrete sampling and straight-through top-k on the answer-preservation task, each with 32 training suffixes and either its default initialization or the local baseline mask. Curves show the median normalized KL on 64 held-out fixed suffixes, aggregated across 19 Qwen3-8B prompts with 95% bootstrap intervals.

E Variation across training seeds

We train masks with seeds 7, 42, and 1234 on one GPQA Diamond and one AQuA-RAT prompt for Qwen3-8B at sparsities 0.5, 0.8, and 0.9. Answer-preservation masks train on 32 suffixes and start from the local baseline; correct-answer masks train on y_{orig} and start from $\phi_{ij} = 2$. The seed changes the hard-concrete samples while the initialization and order of training suffixes remain fixed.

For each pair of seeds, we compare the binary masks M and M' at the same target sparsity s. The shared fraction is the number of connections that both masks keep, divided by the budget k. Two independent random masks share a fraction $k/|\mathcal{E}| \approx 1 - s$ in expectation. We also record absolute differences in training metrics: normalized KL over the 32 training suffixes for answer preservation, and $q_M(a^\star \mid x, y_{\text{orig}})$ for the correct-answer task.

Every seed pair shares more connections than the random-mask expectation (Figure 6). At $s = 0.9$, answer-preservation masks share 44–57% of retained connections and correct-answer masks share 33–48%, compared with a random expectation of about 10%. Thus, 43–67% of the connections kept by one seed are absent from another seed at the same sparsity.

Training performance varies most across seeds at high sparsity. At $s = 0.5$, two seeds differ by at most 0.002 in normalized KL and by at most 0.05 in correct-answer probability. At $s = 0.9$, the normalized KL on the GPQA Diamond prompt ranges from 0.03 to 0.11 across seeds, and the correct-answer probability from 0.76 to 0.86. On the AQuA-RAT prompt, seed 7 reaches correct-answer probabilities of 0.40 at $s = 0.8$ and 0.34 at $s = 0.9$, compared with 0.65–0.73 and 0.55–0.63 for seeds 42 and 1234. Answer-preservation masks at $s = 0.8$ reach the same normalized KL within 0.005 while sharing only 0.62 to 0.67 of their connections. The two-prompt study shows that distinct masks can reach similar training performance, although some seeds also produce worse task scores.

F Datasets and prompt selection

Data collection. We sample 16 reasoning traces per question at temperature 0.6 with vLLM (Kwon et al., 2023). The empirical accuracy is the fraction of traces with the correct answer. The first trace supplies the original reasoning prefix. Both answer tasks use the same selected set of 99 prompts (Table 1); shortening uses 10 prompts. The figure captions specify the subsets covered by each reported evaluation or ablation.

Model	Dataset	Collected	Each answer task	Shortening
Qwen3-8B	GPQA Diamond (Rein et al., 2024)	198	25	5
Qwen3-8B	AQuA-RAT (Ling et al., 2017)	754	17	5
Qwen3-8B	MATH (Hendrycks et al., 2021b)	700	23	–
Qwen3-32B	GPQA Diamond (Rein et al., 2024)	198	20	–
Qwen3-32B	MMLU (Hendrycks et al., 2021a)	800	14	–
Total selected			99	10

Table 1: Questions collected and prompts selected for each task. Collection covers all 198 GPQA Diamond questions, 254 test and 500 training questions from AQuA-RAT, and 40 questions from each of 20 MMLU subjects. Both answer tasks use the same prompts. A dash marks a model–dataset pair outside the shortening study.

Trace requirements. Every original trace must contain `</think>`, a parseable answer, and enough sentences for the task-specific prefix and stored suffix. We split sentences using the rule in Section B.1. The answer tasks keep questions with empirical accuracy strictly between 0.25 and 0.75; shortening uses the separate filter below.

Answer probabilities. The answer set $\mathcal{A}$ contains letters `A`–`D` for GPQA Diamond and MMLU, and `A`–`E` for AQuA-RAT. After the fixed answer prompt, we normalize the unscaled logits over the valid letters, as in Section 3.1. For MATH, we normalize candidate-answer probabilities as described in Section F.1.

Prefix selection for the answer tasks. Both answer tasks use the same prefix for each prompt. We choose the cut using the effect of removing all eligible connections on the original continuation. For each candidate cut we compute $\ell_{\text{preserve}}(\mathbf{0}; x, y_{\text{orig}})$, the KL divergence reached by removing every eligible connection, where y_{orig} holds the five sentences that follow the cut in the original trace. Candidate cuts follow sentence 35, 50, 65, or 80 for Qwen3-8B on GPQA Diamond. The other model–dataset pairs, including Qwen3-8B on MATH, use sentence 20, 35, 50, 65, 80, 100, or 120. The cut indices count sentence spans from the start of the prompt; n in Section 2 counts only reasoning sentences. We keep the cut after sentence 50 when $\ell_{\text{preserve}}(\mathbf{0}; x, y_{\text{orig}}) \geq 0.1$ there, and otherwise the cut with the largest value. We drop prompts whose KL remains below 0.1 at every candidate cut because removing all eligible connections has too little effect for the selection criterion. Answer preservation samples 32 training and 64 held-out fixed suffixes at the chosen cut,

then compares mean answer distributions from 32 generated suffixes and 32 unmasked reference suffixes. The correct-answer task trains on y_{orig} and evaluates the median correct-answer probability over five generated suffixes. All sampled suffixes for the answer tasks use temperature 0.7 and a 200-token limit. The stored suffix inherits the 0.6 sampling temperature of the original trace. For MATH, the selected prompt set excludes questions without an incorrect candidate answer.

Shortening task. The analysis point is the sentence end nearest to 2,200 tokens before the `</think>` of the original trace, or 3,000 tokens before if the 16 continuations sampled at 2,200 tokens all agree on the answer. The analysis point lies at least 10 sentences into the reasoning, and we cap prefixes at 15,000 tokens. We keep prompts whose accuracy over the 16 sampled traces lies in $[0.5, 0.75]$, whose original trace reaches the correct answer, and whose 16 continuations sampled at the analysis point (temperature 1.0) disagree on the final answer. Twelve continuations form $\mathcal{D}_{\text{train}}$; we omit any of the twelve without an eligible break from the loss average. The other four continuations do not enter the reported metrics. Evaluation uses fresh continuations as described in Section B.4.

F.1 Open-ended answers

MATH has no fixed answer choices. For each prompt, we build a candidate bank from 16 completions of the forced opener `\boxed{` at temperature 1.0, incorrect answers from the original 16 traces, and the correct answer. We group exact matches and use an LLM judge to merge equivalent strings. The resulting clusters form $\mathcal{A}$. Let $\mathcal{C}$ be the candidate strings and $r_{\widetilde{M}}(c \mid x, y)$ the product of their token probabilities under unscaled logits after the fixed answer opener. We normalize over the bank and sum within each cluster:

$$q_{\widetilde{M}}(a \mid x, y) = \frac{\sum_{c \in \mathcal{C} \,:\, \text{cluster}(c) = a} r_{\widetilde{M}}(c \mid x, y)}{\sum_{c' \in \mathcal{C}} r_{\widetilde{M}}(c' \mid x, y)}. \tag{18}$$

The losses in Equations (6) and (9) use the cluster probabilities. The distribution covers the candidate bank, so its probabilities are conditional on the included answers.

G Sentence types and judged dependencies

Sentence types. Figure 16 separates the sentence-type analysis by position in the prefix. The learned masks show no consistent preference for one sentence type across objectives and positions, although some types have nonzero log odds ratios within individual bins. The local connection baseline favors planning and uncertainty-management sentences in several bins.

Judged dependencies. The local baseline retains more dependencies marked by the LLM judge than the learned masks (Figure 15). At 80% sparsity, the baseline retains a median 87% of judged dependencies across the pooled prompts and exceeds the learned mask on every prompt in each task. Learned masks retain median fractions of 69% for answer preservation, 23% for the correct-answer task, and 17% for shortening, compared with a random expectation of about 20%. The answer-preservation masks start from the local baseline, but the comparison does not isolate the effect of initialization on dependency retention.

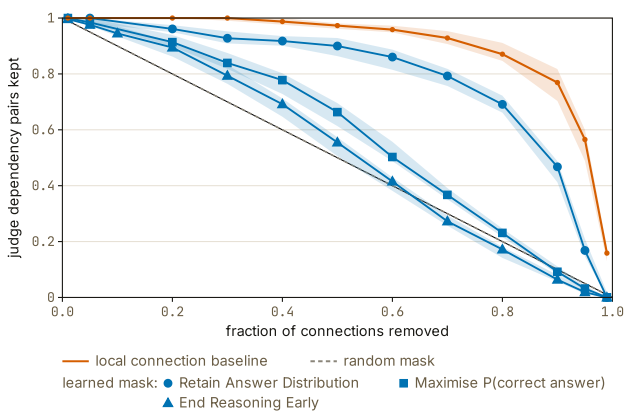


Figure 15: Fraction of judged sentence dependencies retained by each mask. Gemini 2.5 Flash reads the prefix and lists the earlier sentences on which each reasoning sentence depends. Curves show medians across prompts with 95% bootstrap intervals; the dashed line marks the random-mask expectation $1 - s$. The baseline pools 86 distinct model–prompt pairs, and the learned-mask subsets match Figure 7.

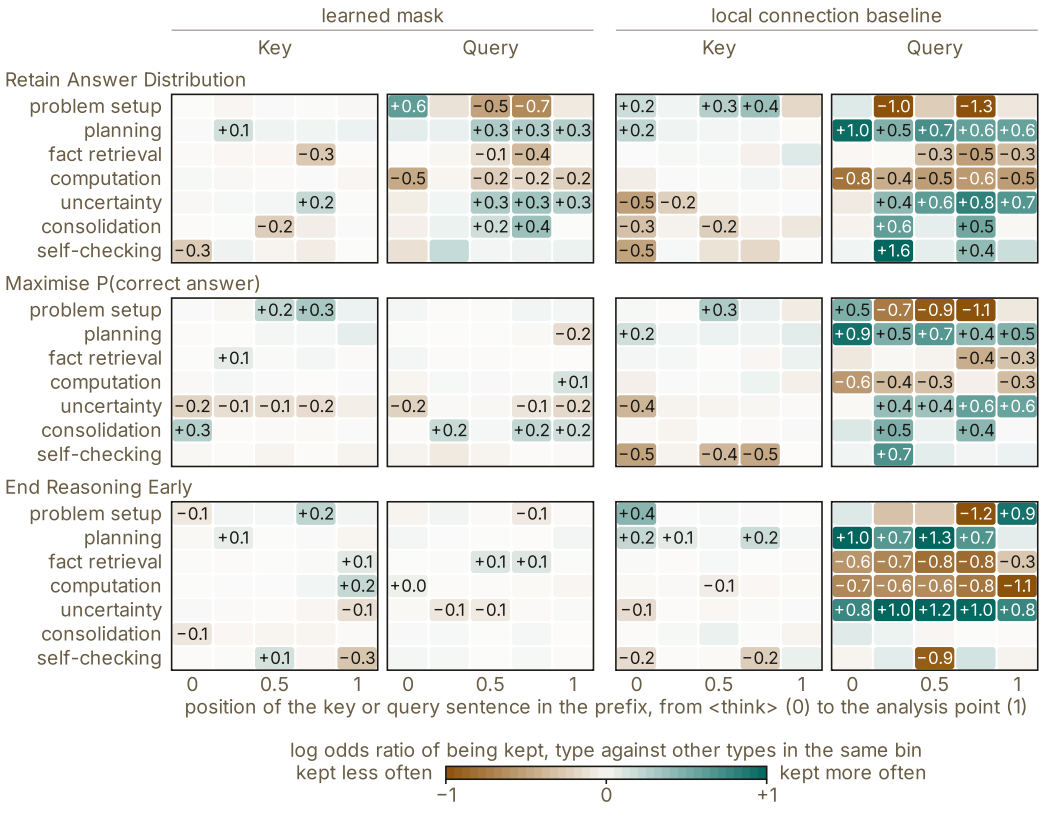


Figure 16: Sentence types of the connections that each mask keeps, by position in the reasoning prefix. An LLM judge assigns categories from Bogdan et al. (2025); we use the first tag for each sentence. Five equal-width token bins span the reasoning prefix from <think> (0) to its end (1). Key and Query columns group connections by the position of the earlier and later sentence, respectively. Each cell shows the log Mantel–Haenszel odds ratio for retaining connections of the indicated type versus other types in the same bin, pooling 2×2 tables for each prompt and sparsity. Zero denotes equal retention odds; pale cells have 95% bootstrap intervals over prompts that include zero. The prompt subsets match Figure 15.