CONSTRUCTIVE CIRCUIT AMPLIFICATION: IMPROVING MATH REASONING IN LLMS VIA TARGETED SUB-NETWORK UPDATES

Nikhil Prakash[1*] **Donghao Ren**[2] **Dominik Moritz**[2] **Yannick Assogba**[2]
[1] Northeastern University
[2] Apple

ABSTRACT

Prior studies investigating the internal workings of LLMs have uncovered sparse subnetworks, referred to as circuits, that are responsible for performing specific tasks. Additionally, it has been shown that performance improvements of fine-tuning often results from the strengthening of existing circuits. Taken together, these findings suggest the possibility of intervening directly on such circuits to make precise, task-targeted updates. Motivated by these findings, we propose a novel method called *Constructive Circuit Amplification* which identifies pivotal tokens from model reasoning traces as well as model components responsible for the desired task, and updates only those components. Applied to mathematical reasoning, it improves accuracy by up to +11.4% across multiple models while modifying as little as 1.59% of model components, with minimal impact on other abilities as measured by MMLU, TriviaQA, and TruthfulQA. These results demonstrate that targeted capabilities can be reliably enhanced by selectively updating a sparse set of model components.

1 INTRODUCTION

Large language models (LLMs) have demonstrated impressive general-purpose reasoning abilities, yet they continue to struggle with mathematical reasoning tasks, where even small logical errors can derail problem-solving (Shojaee et al., 2025; Marjanović et al., 2025; Ballon et al., 2025). Existing works have attempted to improve math reasoning through various prompting and fine-tuning strategies, which have led to modest gains (Wang et al., 2022b; Chen et al., 2022; Lewkowycz et al., 2022; Lightman et al., 2023). In this work, we propose an alternative approach that leverages insights from mechanistic interpretability to achieve more targeted improvements.

Recent progress in mechanistic interpretability has revealed that model behavior is often governed by sparse subnetworks, or circuits, consisting of attention heads and MLP neurons that jointly implement specific capabilities (Wang et al., 2022a; Hanna et al., 2023; Merullo et al., 2023; Prakash et al., 2024; Marks et al., 2025). Jain et al. (2023); Prakash et al. (2024); Chhabra et al. (2025) show that fine-tuning frequently strengthens these existing circuits rather than creating entirely new mechanisms. Additionally, Rai et al. (2025); Ortu et al. (2024) suggest that there is a competition among circuits within a model's internal computation, where some circuits contribute to correct reasoning while others introduce noise. Together, these findings suggest that targeted interventions on circuits could enable precise updates that enhance specific skills while minimizing unrelated disruption.

In this work, we aim to mechanistically understand why LMs produce reasoning errors despite having the latent capacity to solve the underlying problems, and then use these insights to develop a method for correcting them. More specifically, we introduce **Constructive Circuit Amplification** (CCA), a mechanistically informed fine-tuning method that performs sparse, targeted updates to improve LLM reasoning. CCA operates in three stages: (i) generating reasoning traces to identify pivotal tokens where incorrect solutions diverge from correct ones, (ii) localizing the attention heads and MLP neurons that promote correct reasoning paths, and (iii) applying gradient updates exclusively to

*Corresponding author: prakash.nik@northeastern.edu; Work done while on an internship at Apple.

those components. Prior mechanistic interpretability work has largely focused on non-reasoning tasks with single-step outputs, where identifying the responsible components is more straightforward. By amplifying the contribution of the circuits most responsible for correct reasoning, CCA strengthens mathematical reasoning ability while leaving unrelated skills largely intact.

Applied to the GSM-Symbolic benchmark Mirzadeh et al. (2025), CCA yields accuracy improvements of up to +11.4% when tested across multiple model families, while modifying as little as 1.59% of components (i.e. attention heads and MLP neurons). Importantly, these gains come with minimal degradation on general benchmarks including MMLU, TriviaQA, and TruthfulQA, underscoring that targeted improvements can be achieved without compromising broad capabilities (Hendrycks et al., 2021; Joshi et al., 2017; Lin et al., 2022). Although our focus has been on enhancing mathematical reasoning in LLMs, CCA is a modular, plug-and-play framework that can be applied to any model or task with available reasoning traces.

Our results demonstrate that LLM skills can be selectively enhanced by updating only the sparse subnetwork that implements them. By localizing constructive circuits directly from reasoning traces and updating only those components, our method demonstrates that the circuit analysis approach can be extended to reasoning and can also guide sparse, targeted parameter changes that improve reasoning ability while minimizing interference with other skills.

2 Related Works

2.1 Mechanistic Interpretability in LLMs

The field of mechanistic interpretability seeks to reverse-engineer the internal computations of deep neural networks (Olah et al., 2020; Mueller et al., 2024; Saphra & Wiegreffe, 2024). A prominent line of work focuses on uncovering *circuits*, sparse sets of attention heads and MLP neurons that collectively drive specific model behaviors, such as indirect object identification, greater-than, and entity tracking Wang et al. (2022a); Hanna et al. (2023); Prakash et al. (2024). Recent research has also extended this perspective to the sparse feature space, identifying and editing interpretable circuits that govern feature-level interactions Marks et al. (2025); Ameisen et al. (2025).

A recurring theme across this line of work is that LM behavior is not uniformly distributed across parameters, but rather localized within a relatively small subset of components. Jain et al. (2023); Prakash et al. (2024); Chhabra et al. (2025) show that fine-tuning often strengthens existing circuits rather than creating entirely new mechanisms, while (Merullo et al., 2023) highlights how subcircuits are reused across different tasks. These findings motivate our approach of selectively amplifying the circuits responsible for the target task, while minimizing disruption to unrelated capabilities.

RL fine-tuning has been shown to concentrate changes into a small subnetwork, highlighting that only a limited portion of the model often drives behavioral shifts Mukherjee et al. (2025). Separately, Liu et al. (2025) demonstrates that low-rank reduction reveals "principal weights," enabling sparse supervised fine-tuning by updating top-magnitude parameters. Complementing these views, Li et al. (2025) frames fine-tuning as identifying a task-relevant subgraph within the model. While aligned with the premise that only a small subset of components drives task behavior, this perspective differs methodologically from our approach, which uses behavior-guided mechanistic localization, via reasoning-trace divergence and DCM, to pinpoint and selectively amplify the specific circuits underlying correct mathematical reasoning.

2.2 Mathematical Reasoning with LLMs

Improving mathematical reasoning in LLMs has been a central challenge, as even minor logical mistakes can derail otherwise promising problem-solving attempts (Wang et al., 2025). One line of research has focused on prompting strategies, such as chain-of-thought prompting, self-consistency, and program-of-thoughts prompting, which encourage models to externalize intermediate steps and thereby improve reliability (Wang et al., 2022b; Chen et al., 2022; Lightman et al., 2023). Another line of work investigates fine-tuning techniques, including supervised fine-tuning on reasoning traces or parameter-efficient approaches like LoRA, which can adapt models toward stronger mathematical reasoning (Lewkowycz et al., 2022).

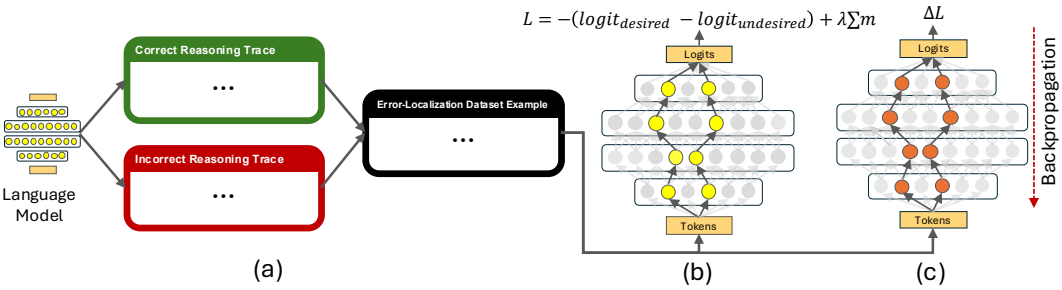

Figure 1: **Overview of CCA**: (a) *Token Localization*: For a given problem, we generate both correct and incorrect reasoning traces and identify the pivotal token where the incorrect trace diverges from the correct one. The intervention point is chosen as the token immediately preceding this divergence. (b) *Model Component Localization*: Using the Error-Localization dataset constructed from these reasoning trace pairs, we apply DCM to learn a sparse binary mask over attention heads and MLP neurons. This identifies the subset of components that most strongly promote the desired token. (c) *Model Update*: Gradient updates are then applied exclusively to the localized components, amplifying constructive computations while leaving the rest of the network unchanged.

Recent research has also examined the internal mechanisms of LLMs to better understand their mathematical reasoning capabilities. For example, Ye et al. (2024) analyzed the internal activations of a transformer model trained from scratch on a math reasoning dataset, using probes to uncover mechanisms underlying the reasoning ability. Similarly, Sun et al. (2025b) trained probes to predict the correctness of outputs in 3-digit addition, showing strong generalization to addition-only GSM8K problems. By leveraging these probes, they selectively re-prompted erroneous reasoning steps, thereby improving task accuracy. A closely related study, Sun et al. (2025a), introduced ThinkEdit, which identifies attention heads responsible for short reasoning traces and updates their weights to extend these traces, ultimately enhancing model performance. Building on this line of work, we show that localization-informed, targeted sub-network updates can strengthen mathematical capabilities.

3 Constructive Circuit Amplification

3.1 Method Overview

We propose a novel technique, called *Constructive Circuit Amplification*, to improve the mathematical reasoning capabilities of an LM, without affecting other abilities. The underlying premise of this method relies on two empirical insights from the mechanistic interpretability literature: 1) Specific tasks in LM are often executed by a sparse subnetwork, which gets augmented during fine-tuning, leading to model performance improvement (Jain et al., 2023; Prakash et al., 2024; Chhabra et al., 2025). 2) There is a competition among various mechanisms within an LM's internal computation, some of which are sound for the given task, while others are introducing noise, as suggested in Rai et al. (2025); Ortu et al. (2024). Mirzadeh et al. (2025) shows that even when models have a decent ability to solve math reasoning tasks; on certain instances, they produce incorrect intermediate outputs and thus incorrect final answers. To overcome this shortcoming, CCA amplifies the signal from model components that are constructively support generating the correct response. The technique consists of three steps: 1) Generation of Error-Localization dataset, 2) Training binary mask to localize constructive model components, and 3) Updating only those model components using a few gradient update steps. The following subsections describe each step in more detail.

3.1.1 Localizing Reasoning Errors

The first step of CCA identifies the point in the reasoning trace where the model begins to deviate toward an incorrect answer, as illustrated in Fig. 1(a). Prior work on circuit discovery has primarily examined tasks where the output is produced *in a single forward pass*, such as indirect object identification, entity tracking, and greater-than comparison (Wang et al., 2022a; Prakash et al., 2024; Hanna et al., 2023), making it natural to apply circuit discovery methods at the final token position. In contrast, mathematical reasoning involves multi-step computations, and it is less clear at which point to apply the circuit discovery algorithm to uncover the circuit that can be enhanced to improve

overall reasoning ability. To address this, we begin circuit localization by identifying the token in the reasoning trace where an intervention should occur. Specifically, for a GSM-Symbolic instance where the model produces an incorrect solution through greedy sampling, we aim to locate the first token in the reasoning trace that drives the model toward this error. We refer to this token as the *pivotal token*. Our intervention then targets the token immediately preceding it, to discourage the model from generating the pivotal token. We refer to this intervention point, where we amplify signals from constructive model components, as the *intervention token*.

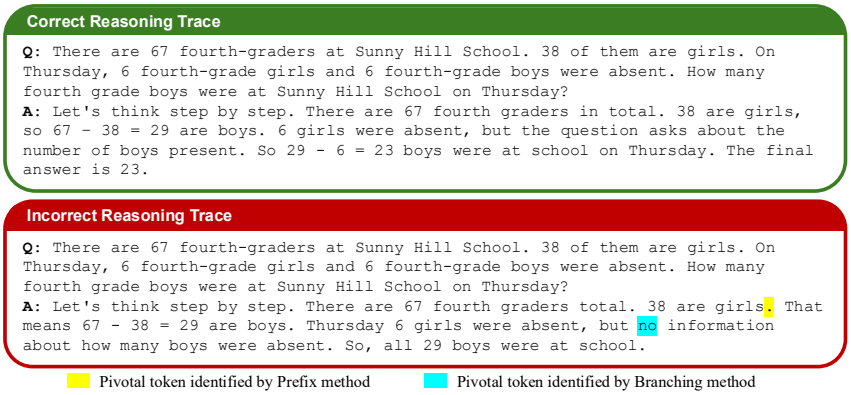

Figure 2: Example of a GSM-Symbolic math word problem showing both a correct and an incorrect reasoning trace produced by `Gemma-2-9b-Instruct` model. The correct trace (top) is obtained through greedy decoding, while the incorrect trace (bottom) is produced by non-greedy sampling.

For each GSM-Symbolic instance, we first sample a reasoning trace and final answer using greedy decoding. If the answer is incorrect, we generate an alternative reasoning trace that leads to the correct answer via non-greedy decoding. Conversely, if greedy decoding produces the correct answer, we instead generate an incorrect reasoning trace using non-greedy decoding. With this paired set of reasoning traces, we then apply one of the following methods to identify the intervention token.

Prefix Method: Given a pair of reasoning traces, this method identifies the first token that is not shared between them as the pivotal token. For instance, in the Fig. 2, the first uncommon token between both the reasoning traces is the "," and "." tokens. Consequently, its prior token, i.e. "$girls$", becomes the intervention token.

Although efficient, this method sometimes identify suboptimal pivotal and intervention tokens. For example, in Fig. 2, the first differing tokens in the two traces are "," and ".". However, these tokens are not the decisive points that steer the model toward a correct or incorrect reasoning path. Specifically, when the reasoning trace "`There are 67 fourth graders total. 38 are girls.`" is provided as input, the model still produces the correct final answer via greedy sampling. This shows that the "." token is not a decisive token.

Branching Method: To address this challenge, we propose a method based on *iterative greedy decoding with partial prefixes*. Suppose we have a correct reasoning trace (i.e. token sequence) $T^{\text{corr}} = (x_1, x_2, \ldots, x_n)$, obtained via greedy decoding, and an incorrect reasoning trace $T^{\text{incorr}} = (y_1, y_2, \ldots, y_m)$, obtained via non-greedy decoding. Our goal is to identify the pivotal token in T^{incorr} that steers the model toward an incorrect final answer. Formally, let $f(\cdot)$ denote the final answer of a reasoning trace generated via greedy decoding, and let $\mathcal{A}_{\text{corr}}$ and $\mathcal{A}_{\text{incorr}}$ denote the correct and incorrect final answers, respectively. Then a token y_k is defined as pivotal if $f(y_1, \ldots, y_{k-1}) \in \mathcal{A}_{\text{corr}}$ and $f(y_1, \ldots, y_k) \in \mathcal{A}_{\text{incorr}}$.

Operationally, we construct a prefix of length k from T^{incorr}, i.e., $(y_1, \ldots, y_k)$, and feed it into the model to complete the reasoning trace using greedy decoding. We then check whether the resulting final answer is correct. If it is correct, we extend the prefix by adding the next token y_{k+1} and repeat the procedure. If the final answer is incorrect, then the newly added token y_k is identified as the pivotal token, since its inclusion causes greedy decoding to lead to an incorrect outcome. In the example shown in Fig. 2, it is the "no" token which pushes the model trajectory towards an incorrect final answer. Hence, it becomes the pivotal token.

In the opposite case, when greedy decoding yields an incorrect reasoning trace while non-greedy decoding yields a correct one, we apply the same procedure. The difference is that the pivotal token is now defined as the first token in the non-greedy trace whose inclusion in the prefix causes greedy decoding to switch from an incorrect to a correct final answer. In this case, y_k is pivotal if $f(y_1, \ldots, y_{k-1}) \in \mathcal{A}_{\text{incorr}}$ and $f(y_1, \ldots, y_k) \in \mathcal{A}_{\text{corr}}$.

Error-Localization Dataset Example

```
prefix: There are 67 fourth graders in total. 38 are
girls, so 67 - 38 = 29 are boys. 6 girls were absent, but
desired_token: the
undesired_token: no
```

Figure 3: The Error-Localization dataset contains three components: *prefix*: the shared reasoning trace between the correct and incorrect paths (including intervention token), *desired_token*: the token the model should generate to produce the correct answer, and *undesired_token*: the token the model should avoid generating to ensure the correct answer.

Finally, after identifying the intervention token and corresponding pair of reasoning traces for a given GSM-Symbolic instance, we construct the *Error-Localization* dataset. As illustrated in Fig. 3 each instance in this dataset consists of three components: 1) *Prefix*: the shared reasoning trace up to and including the intervention token, 2) *Desired token*: the token following the intervention token in the correct reasoning trace, and 3) *Undesired token*: the token following the intervention token in the incorrect reasoning trace.

3.1.2 Identifying Constructive Circuits with DCM

Using the training dataset generated in the previous step, we can localize errors in an incorrect reasoning trace to specific tokens. Next, we go further and identify the model components responsible for promoting the correct reasoning trace, or more specifically, the generation of the desired token, as shown in Fig. 1(b). To achieve this, we leverage the Desiderata-based Component Masking (DCM) technique (Davies et al., 2023; De Cao et al., 2022; Prakash et al., 2024; 2025).

DCM learns a binary mask over key, query, and value weight matrices of all attention heads and MLP neurons in the LM by minimizing a tailored loss function. More specifically, it learns $n\_heads + 2 * n\_key\_value\_heads + n\_mlp\_neurons$ parameters for each layer, where n_heads represents the number of attention heads, $n\_key\_value\_heads$ represents the number of key and value heads in Grouped Attention, and $n\_mlp\_neurons$ represents the number of MLP neurons. Each parameter in the mask represents whether its corresponding model component should be intervened on or left unchanged during the forward pass. We use the following equation to update a model component's output using the mask:

$$h_{org} = m_i * 2 * h_{org} + (1 - m_i) * h_{org} \tag{1}$$

where h_{org} represents the original model component output and m_i represents the corresponding mask value. Concretely, if a component's mask value is 1, its activation is scaled by 2; otherwise, it remains unchanged. It is implemented using NNsight (Fiotto-Kaufman et al., 2024).

Since our goal is to isolate the components that promote the desired token while suppressing the undesired one, we use a loss function defined as the logit difference between the undesired and desired tokens to optimize the binary mask. To encourage sparsity in the mask, we add an L_1-norm regularization term, weighted by the hyperparameter λ. It ensures that only a small subset of components is identified as influential. Formally, the loss is:

$$\mathcal{L} = -(\text{logit}_{\text{desired\_token}} - \text{logit}_{\text{undesired\_token}}) + \lambda \sum \mathbf{m} \tag{2}$$

where λ controls the sparsity of the binary mask, and its optimal value is selected by sweeping over a range of candidate values. We report the percentage of mask components in the results section, computed as $\frac{|M_{\text{learned}}|}{|M|}$, where $|M_{\text{learned}}|$ denotes the number of components selected by the learned mask and $|M|$ is the total number of components in the mask. The distribution of selected components across Q, K, V heads, and MLP neurons is reported in Table 7.

We train the binary mask using the Adam optimizer for 50 epochs, with a learning rate of 5e-3, batch size of 8, and tuning the λ via parameter sweeps. Full details are in App. C. To prevent unnecessary computation, we apply early stopping: if the mask remains unchanged after 20% of batches in an epoch (i.e., the set of selected components does not vary), training is halted. Additionally, after each gradient update, we clamp the mask values to the range $[0, 1]$, as values outside this interval are incompatible with Eq. 1. In summary, the learned mask identifies a small set of model components, *the circuit*, whose **amplified outputs** steer the model toward the correct reasoning trace.

3.1.3 Targeted Parameter Updates

After identifying the model components that promote the desired token, we update only these components using gradient descent, as illustrated in Fig. 1(c). We use the negative logit difference between the desired and undesired tokens as the loss function from the Error-Localization dataset. Gradients are applied exclusively to the previously identified components. Because the training dataset is small and only a limited number of updates are expected, we compute gradients over the entire dataset rather than using mini-batches. We perform a total of 50 gradient update steps, evaluating the model's exact match accuracy on the validation set every 2 steps up to step 10, and subsequently every 10 steps. At the end of training, we select the best-performing updated model and evaluate it on the test set and report the results in Fig. 5. The optimal learning rate is determined through a sweep over candidate values[1].

4 Experimental Setup

4.1 Datasets for Math and General Abilities

We evaluate the effectiveness of CCA on improving the mathematical reasoning capabilities of LMs while preserving other skills gained during pretraining. For math reasoning, we use the GSM-Symbolic (Mirzadeh et al., 2025) benchmark, which provides templates derived from the GSM8K dataset (Cobbe et al., 2021). The benchmark contains 100 math problem templates across diverse topics, each with 50 instances. We randomly divide these instances into training, validation, and test sets in proportions of 0.52, 0.08, and 0.40, respectively. Thus, for each of the 100 templates, there are 26 training, 4 validation, and 20 test instances. We further filter our train split *to only include templates whose mean accuracy is below* 0.8 on the target model. A full list of selected templates is provided in App. B. In the rest of this paper, we refer to these splits as **GSym-Train**, **GSym-Val**, and **GSym-Test** respectively.

It is important to note that the model does not always produce a counterfactual reasoning trace under non-greedy sampling. Consequently, some GSM-Symbolic instances are absent from the Error-Localization dataset for the prefix and branching methods. As a result, the size of the training dataset varies across models and localization generation types, and it is always smaller than the maximum of 2600. Table 1 reports the training set sizes for all models considered. The validation and test sets consistently contain 400 and 2000 instances, respectively, from the original GSM-Symbolic.

In addition to mathematical reasoning, we also evaluate the general capabilities of LMs using the MMLU, TriviaQA, and TruthfulQA benchmarks (Hendrycks et al., 2021; Joshi et al., 2017; Lin et al., 2022). MMLU includes questions spanning a broad range of topics. To better assess any unintended effects of enhancing math reasoning, a skill central to many STEM tasks, we evaluate on two MMLU subsets: "MMLU Stem" and "MMLU Humanities" as defined within MMLU. A complete list of both STEM and Humanities categories is provided in App. D.

4.2 Evaluated Model Families

We evaluate CCA across multiple families of open-weight LLMs to assess its robustness and generality, namely Gemma and OLMo (Team et al., 2024; Groeneveld et al., 2024). Specifically, we analyse `Gemma-2-9b-Instruct`, `Gemma-2-2b-Instruct`, `OLMo-2-1124-13B-Instruct`, and `OLMo-2-1124-7B-Instruct` models. Our manual inspection of erroneous reasoning traces of these models reveals that most errors stem from failures in logical reasoning steps rather than from

[1]Candidate learning rates: 1e-2, 5e-3, 1e-3, 5e-4, 1e-4, 5e-5, 1e-5.

arithmetic mistakes. For example, Fig. 2 illustrates how `Gemma-2-9b-Instruct` produces an incorrect answer to a GSM-Symbolic instance due to its inability to extract the necessary information from the question, rather than an arithmetic error.

4.3 Baseline: LoRA Fine-Tuning

We compare our method against LoRA fine-tuning, a well-established parameter-efficient fine-tuning method (Hu et al., 2022) often used for task-adaptation. We use the same **GSym-Train** data splits used for CCA as described in Section 4.1. We did not train LoRA on the Error-Localization dataset because that dataset is a core component of CircuitTuning itself and using it would blur the comparison between methods. Moreover, the localization dataset is far smaller than LoRA's standard training split, which would severely undertrain LoRA and yield an unfairly weakened baseline. We apply LoRA to both the attention and MLP components of each transformer block, and run finetuning with an effective batch size of 32 for two epochs. We evaluate the model on the validation set every 10 steps with the same exact-match metric used for CCA and select the best-performing model checkpoint. We also sweep over a number of learning rates and report the best performing results on the test set (**GSym-Test**) in Section 5. Other LoRA hyperparameters, such as rank, learning rate schedule, etc, are fixed for all models and complete details can be found in App. H.1.

5 Experimental Results on Math Reasoning

This section presents the results of the original unmodified models, the models updated with CCA, and LoRA finetuned models on **GSym-Test** (CCA results are averaged over three random seeds). For each localization dataset generation type, we report two configurations: (1) *CCA w/ mask*, where only the model components identified by the mask are updated, and (2) *CCA w/o mask*, an ablation where we skip model component localization via DCM and allow any model component to be updated during the gradient update step. The purpose of reporting both configurations is to disentangle the effects of model component localization and reasoning token localization.

Table 1 presents the results, highlighting a few key observations. First, models updated with CCA show substantially better performance than their corresponding base models across multiple model families. The improvement can be as high as 12.1% (for `Gemma-2-2b-Instruct`) using only 1244 samples. Moreover, CCA surpasses LoRA, a strong baseline, in both the `Gemma-2-9b-Instruct` and `OLMo-2-1124-7B-Instruct` models. These results indicate that CCA can fix math reasoning failures substantially through sparse, mechanism-aligned updates, hence could be an effective option for capability improvement, particularly in data-constrained settings where maximizing performance from limited samples is critical.

Second, we find that models updated with the Branching localization dataset consistently outperform those updated with the Prefix method across model families. This suggests that accurately identifying the pivotal reasoning token that steers the model toward incorrect reasoning is critical for improving performance. More broadly, it underscores the importance of developing improved techniques for localizing token(s) within long reasoning traces, as a means of uncovering the underlying circuits and mechanisms responsible for different reasoning tasks.

Finally, we observe that the number of attention heads and MLP neuron weights that need to be updated to improve performance constitutes only a small fraction of the total model components. For example, achieving a 7.4% performance gain in `Gemma-2-9b-Instruct` requires updating only 0.17% of its components. This suggests that a limited subset of model components is primarily responsible for correct reasoning on the GSM-Symbolic dataset, and amplifying their contribution can significantly enhance performance. Moreover, since CCA updates only a small fraction of the model, we expect minimal interference with other capabilities acquired during pre-training.

6 Preserving Broader LM Abilities

Results in Table 1 show that CCA is effective in improving task performance for multiple models. To understand potential side-effects on broader model capabilities, we use LM-evaluation-harness (Gao et al., 2024) to evaluate the updated LMs on MMLU, TriviaQA, and TruthfulQA (Hendrycks et al., 2021; Joshi et al., 2017; Lin et al., 2022). We compare the best updated model with CCA against the

Table 1: Performance comparison of CCA and LoRA fine-tuning across multiple models on the GSM-Symbolic benchmark. We report accuracy on the **GSym-Test** under different training configurations: (i) CCA with a mask (updates restricted to components identified by the learned mask), (ii) CCA without a mask (updates applied more broadly), and (iii) LoRA fine-tuning. % Mask is the percentage of model components updated and Δ% Acc is the absolute accuracy improvement

Configuration	Dataset	Dataset Size	% Mask	GSym-Test Acc	Std	Δ % Acc
Gemma-2-9B-Instruct						
Original Model	–	–	–	0.807	–	–
CCA w mask	Prefix	510	0.13%	0.848	±0.006	4.1
CCA w/o mask	Prefix	510	–	0.849	±0.011	4.2
CCA w mask	Branching	512	0.17%	0.881	±0.015	**7.4**
CCA w/o mask	Branching	512	–	0.875	±0.010	6.8
LoRA Finetuning	**GSym-Train**	676	–	0.850	–	4.3
Gemma-2-2B-Instruct						
Original Model	–	–	–	0.411	–	–
CCA w mask	Prefix	1,283	0.92%	0.440	±0.011	2.9
CCA w/o mask	Prefix	1,283	–	0.502	±0.018	9.1
CCA w mask	Branching	1,244	1.59%	0.525	±0.010	11.4
CCA w/o mask	Branching	1,244	–	0.532	±0.009	12.1
LoRA Finetuning	**GSym-Train**	2,028	–	0.579	–	**16.8**
OLMo-2-1124-13B-Instruct						
Original Model	–	–	–	0.742	–	–
CCA w mask	Prefix	864	0.37%	0.768	±0.018	2.6
CCA w/o mask	Prefix	864	–	0.762	±0.002	2.0
CCA w mask	Branching	845	0.44%	0.786	±0.005	4.4
CCA w/o mask	Branching	845	–	0.784	±0.006	4.2
LoRA Finetuning	**GSym-Train**	1,118	–	0.797	–	**5.5**
OLMo-2-1124-7B-Instruct						
Original Model	–	–	–	0.739	–	–
CCA w mask	Prefix	974	0.19%	0.772	±0.018	3.3
CCA w/o mask	Prefix	974	–	0.777	±0.006	3.8
CCA w mask	Branching	983	0.25%	0.794	±0.006	5.5
CCA w/o mask	Branching	983	–	0.806	±0.012	**6.7**
LoRA Finetuning	**GSym-Train**	1,222	–	0.746	–	0.7

original model, as well as a LoRA finetuned model to assess the impact of these updates on general performance. As described in Section 4.1, we report two values for the MMLU benchmark: the mean accuracy over STEM topics and over Humanities topics.

Table 2 shows that models updated with CCA achieve performance comparable to the base model on various standard benchmarks, suggesting that they retain most of the capabilities acquired during pretraining. The combination of targeted accuracy gains with minimal degradation highlights CCA as a safe and effective method for fine-tuning, particularly in applications where retaining broad competencies is essential.

7 Discussion and Conclusion

We introduce CCA, a targeted model update method for amplifying specific model capabilities while preserving general performance. Unlike conventional finetuning strategies that update a large number of model components, CCA identifies pivotal reasoning errors and the circuit responsible for correct reasoning, then applies updates only to those components. Our experiments demonstrate that CCA substantially improves mathematical reasoning across multiple model families, up to +11.4%

Table 2: Absolute percentage difference (in 0–100 scale) between original model and updated model for four CCA conditions and LoRA. Results are shown over five benchmarks: **GSym-Test**, MMLU Humanities, MMLU STEM, TriviaQA, and TruthfulQA.

Configuration	GSym-Test	MMLU Humanities	MMLU STEM	TriviaQA	TruthfulQA
Gemma-2-9B-Instruct					
Prefix w mask	4.1	0.1	0.6	-0.4	0.0
Prefix w/o mask	4.2	-0.4	0.6	-4.0	0.1
Branching w mask	**7.4**	0.2	0.8	2.0	-0.3
Branching w/o mask	6.8	0.2	0.7	0.0	-0.3
LoRA	4.3	0.3	0.5	1.9	-0.1
Gemma-2-2B-Instruct					
Prefix w mask	2.9	0.2	0.1	0.8	-0.7
Prefix w/o mask	9.1	-0.6	0.8	-1.3	-1.6
Branching w mask	11.4	0.0	0.3	1.3	0.1
Branching w/o mask	12.1	0.3	0.6	1.6	0.7
LoRA	**16.8**	-1.0	0.5	0.5	-2.0
OLMo-2-1124-13B-Instruct					
Prefix w mask	2.6	0.1	0.1	-0.4	-0.1
Prefix w/o mask	2.0	0.4	0.0	-0.0	-0.3
Branching w mask	4.4	0.1	0.1	-0.6	-0.3
Branching w/o mask	4.2	0.0	-0.2	-0.5	-0.5
LoRA	**5.5**	0.4	0.2	1.7	0.1
OLMo-2-1124-7B-Instruct					
Prefix w mask	3.3	-0.5	-0.4	-0.0	-0.0
Prefix w/o mask	3.8	0.1	-0.3	-0.3	0.8
Branching w mask	5.5	-0.1	-0.1	-0.1	0.1
Branching w/o mask	**6.7**	0.4	0.2	-0.6	2.0
LoRA	0.7	-0.1	-0.3	0.4	-0.2

accuracy gain, while modifying as little as 1.59% of components. Importantly, these improvements come with minimal degradation on general-purpose benchmarks such as MMLU, TriviaQA, and TruthfulQA. These findings suggest that model capabilities are often governed by sparse, localized subnetworks that can be selectively strengthened to achieve reliable skill amplification.

Beyond improving mathematical reasoning, CCA highlights a broader principle: mechanistically informed, sparse updates provide a pathway to safe and effective model adaptation. This offers practical benefits for real-world deployment, where users expect improvements in targeted abilities without unexpected trade-offs, and contributes to the growing intersection between parameter-efficient finetuning and interpretability-guided model editing.

There are, however, limitations. We focused on mathematical reasoning as a testbed, leaving open whether similar gains can be achieved for other complex domains such as code generation or scientific problem solving. The current approach requires constructing error-localization datasets for token localization, which may be costly in settings without well-defined correctness signals. Further, our experiments considered only a single fine-tuning stage, whereas deployed models often undergo multiple rounds of fine-tuning to enhance different capabilities. However, in such continual learning settings, conventional techniques may introduce substantial regressions (Scialom et al., 2022; Luo et al., 2025), making CCA a promising alternative.

Future work could address these limitations by (i) extending CCA to other capabilities and domains, such as code generation, scientific reasoning, or multi-modal tasks, thereby testing its generality beyond symbolic math; (ii) speeding-up the error-localization process using frontier LLMs to reduce reliance on multiple generations and improve scalability to datasets with longer reasoning traces; and (iii) incorporating more sophisticated optimization techniques to refine model components more efficiently than naive gradient descent.

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.

Marthe Ballon, Andres Algaba, and Vincent Ginis. The relationship between reasoning and performance in large language models – o3 (mini) thinks harder, not longer, 2025. URL https://arxiv.org/abs/2502.15631.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Vishnu Kabir Chhabra, Ding Zhu, and Mohammad Mahdi Khalili. Neuroplasticity and corruption in model mechanisms: A case study of indirect object identification. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3099–3122, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.170. URL https://aclanthology.org/2025.findings-naacl.170/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

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

Nicola De Cao, Leon Schmid, Dieuwke Hupkes, and Ivan Titov. Sparse interventions in language models with differentiable masking. In Jasmijn Bastings, Yonatan Belinkov, Yanai Elazar, Dieuwke Hupkes, Naomi Saphra, and Sarah Wiegreffe (eds.), *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 16–27, Abu Dhabi, United Arab Emirates (Hybrid), December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.blackboxnlp-1.2. URL https://aclanthology.org/2022.blackboxnlp-1.2/.

Jaden Fiotto-Kaufman, Alexander R Loftus, Eric Todd, Jannik Brinkmann, Koyena Pal, Dmitrii Troitskii, Michael Ripa, Adam Belfki, Can Rager, Caden Juang, et al. Nnsight and ndif: Democratizing access to open-weight foundation model internals. *arXiv preprint arXiv:2407.14561*, 2024.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models, 2024. URL https://arxiv.org/abs/2402.00838.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model, 2023. URL https://arxiv.org/abs/2305.00586.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Samyak Jain, Robert Kirk, Ekdeep Singh Lubana, Robert P Dick, Hidenori Tanaka, Edward Grefenstette, Tim Rocktäschel, and David Scott Krueger. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. *arXiv preprint arXiv:2311.12786*, 2023.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Yueyan Li, Wenhao Gao, Caixia Yuan, and Xiaojie Wang. Fine-tuning is subgraph search: A new lens on learning dynamics, 2025. URL https://arxiv.org/abs/2502.06106.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. URL https://arxiv.org/abs/2109.07958.

Zihang Liu, Tianyu Pang, Oleg Balabanov, Chaoqun Yang, Tianjin Huang, Lu Yin, Yaoqing Yang, and Shiwei Liu. Lift the veil for the truth: Principal weights emerge after rank reduction for reasoning-focused supervised fine-tuning, 2025. URL https://arxiv.org/abs/2506.00772.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning, 2025. URL https://arxiv.org/abs/2308.08747.

Sara Vera Marjanović, Arkil Patel, Vaibhav Adlakha, Milad Aghajohari, Parishad BehnamGhader, Mehar Bhatia, Aditi Khandelwal, Austin Kraft, Benno Krojer, Xing Han Lù, Nicholas Meade, Dongchan Shin, Amirhossein Kazemnejad, Gaurav Kamath, Marius Mosbach, Karolina Stańczak, and Siva Reddy. Deepseek-r1 thoughtology: Let's think about llm reasoning, 2025. URL https://arxiv.org/abs/2504.07128.

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, 2025. URL https://arxiv.org/abs/2403.19647.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Circuit component reuse across tasks in transformer language models. *arXiv preprint arXiv:2310.08744*, 2023.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, 2025. URL https://arxiv.org/abs/2410.05229.

Aaron Mueller, Jannik Brinkmann, Millicent Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, et al. The quest for the right mediator: A history, survey, and theoretical grounding of causal interpretability. *arXiv preprint arXiv:2408.01416*, 2024.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models, 2025. URL https://arxiv.org/abs/2505.11711.

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.

Francesco Ortu, Zhijing Jin, Diego Doimo, Mrinmaya Sachan, Alberto Cazzaniga, and Bernhard Schölkopf. Competition of mechanisms: Tracing how language models handle facts and counterfactuals. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8420–8436, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.458. URL https://aclanthology.org/2024.acl-long.458/.

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.

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. *arXiv preprint arXiv:2505.14685*, 2025.

Daking Rai, Samuel Miller, Kevin Moran, and Ziyu Yao. Failure by interference: Language models make balanced parentheses errors when faulty mechanisms overshadow sound ones, 2025. URL https://arxiv.org/abs/2507.00322.

Naomi Saphra and Sarah Wiegreffe. Mechanistic? *arXiv preprint arXiv:2410.09087*, 2024.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6107–6122, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.410. URL https://aclanthology.org/2022.emnlp-main.410/.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity, 2025. URL https://arxiv.org/abs/2506.06941.

Chung-En Sun, Ge Yan, and Tsui-Wei Weng. Thinkedit: Interpretable weight editing to mitigate overly short thinking in reasoning models. *arXiv preprint arXiv:2503.22048*, 2025a.

Yucheng Sun, Alessandro Stolfo, and Mrinmaya Sachan. Probing for arithmetic errors in language models. *arXiv preprint arXiv:2507.12379*, 2025b.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine

Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

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, 2022a. URL https://arxiv.org/abs/2211.00593.

Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Yi-Di Wang, Shu Yan, Cheng-Xing Jia, Xu-Hui Liu, Xin-Wei Chen, Jia-Cheng Xu, Ziniu Li, et al. A survey on large language models for mathematical reasoning. *arXiv preprint arXiv:2506.08446*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022b.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. *arXiv preprint arXiv:2407.20311*, 2024.

A THE USE OF LARGE LANGUAGE MODELS (LLMS)

We used LLMs as a writing assistant to correct grammatical and typographical errors; beyond this, they did not contribute to any stage of the research.

B TEMPLATES UTILIZED IN ERROR-LOCALIZATION DATASET GENERATION

Table 3: Template IDs used for training across different models.

Model	**Template IDs**
`gemma-2-9b-it`	520, 364, 116, 184, 984, 1247, 480, 266, 20, 410, 43, 1207, 456, 989, 357, 1133, 1165, 434, 406, 1239, 858, 1088, 1021, 39, 652, 976
`gemma-2-2b-it`	520, 1305, 164, 991, 740, 103, 491, 364, 365, 496, 116, 125, 1025, 1031, 1020, 145, 401, 788, 918, 921, 158, 930, 1189, 1063, 184, 440, 458, 718, 728, 984, 473, 1247, 480, 636, 1277, 1026, 265, 266, 11, 20, 410, 1053, 800, 546, 937, 43, 1207, 1084, 320, 456, 982, 989, 99, 357, 1133, 1141, 737, 554, 1165, 1264, 304, 242, 434, 336, 661, 406, 1239, 858, 955, 1088, 459, 1021, 39, 74, 107, 652, 1164, 976
`OLMo-2-1124-7B-Instruct`	520, 1305, 991, 103, 873, 116, 1031, 1020, 145, 788, 184, 728, 984, 1247, 1275, 636, 1026, 265, 266, 11, 20, 410, 1053, 800, 43, 1207, 320, 989, 99, 357, 1133, 554, 1165, 1264, 304, 242, 434, 336, 406, 1239, 858, 1088, 459, 1021, 39, 652, 976
`OLMo-2-1124-13B-Instruct`	520, 1305, 300, 991, 740, 103, 364, 116, 184, 728, 984, 473, 1247, 1275, 1026, 265, 266, 11, 20, 410, 1053, 43, 1207, 1084, 320, 989, 99, 357, 1133, 554, 1264, 304, 434, 406, 1239, 858, 1088, 459, 39, 74, 107, 652, 976

C DCM TUNING DETAILS

Table 4: Hyperparameters used for Desiderata-based Component Masking (DCM) tuning. Values include training schedule, optimizer settings, and λ sweep for sparsity control.

Category	**Hyperparameter**	**Value**
Training schedule	Base learning rate	5e-3
	Epochs	50
	Effective batch size	8
Optimization	Optimizer	Adam
	β	0.9
	λ	{1e-2, 1e-3, 5e-3, 1e-4}

D MMLU CATEGORIES

D.1 MMLU STEM TASKS

mmlu_abstract_algebra, mmlu_anatomy, mmlu_astronomy, mmlu_college_biology, mmlu_college_chemistry, mmlu_college_computer_science, mmlu_college_mathematics, mmlu_college_physics, mmlu_computer_security, mmlu_conceptual_physics, mmlu_electrical_engineering, mmlu_elementary_mathematics, mmlu_high_school_biology, mmlu_high_school_chemistry, mmlu_high_school_computer_science, mmlu_high_school_mathematics, mmlu_high_school_physics, mmlu_high_school_statistics, mmlu_machine_learning

D.2 MMLU HUMANITIES TASKS

mmlu_formal_logic, mmlu_high_school_european_history, mmlu_high_school_us_history, mmlu_high_school_world_history, mmlu_international_law, mmlu_jurisprudence, mmlu_logical_fallacies, mmlu_moral_disputes, mmlu_moral_scenarios, mmlu_philosophy, mmlu_prehistory, mmlu_professional_law, mmlu_world_religions

E POTENTIAL RISKS

Our method enables targeted amplification of specific model capabilities, which could be misused to selectively enhance undesirable behaviors if applied without appropriate safeguards. While we observe minimal impact on general-purpose benchmarks, sparse circuit-level updates may still introduce subtle or long-tail behavioral changes that are not captured by standard evaluations. Additionally, the approach relies on reasoning traces and correctness signals, which may be less well-defined or introduce bias in domains without objective ground truth. We mitigate these risks by focusing on mathematical reasoning tasks with clear correctness criteria and by evaluating effects on multiple downstream benchmarks.

F DATA PRIVACY

This work does not involve the collection of new human-generated data. All experiments use established, publicly available benchmarks for mathematical reasoning and general evaluation (e.g., GSM-Symbolic, MMLU, TriviaQA, TruthfulQA), which are designed to avoid personally identifying information and offensive content. GSM-Symbolic consists of templated, synthetic math word problems, and the other benchmarks are widely used and curated datasets with documented data collection and filtering procedures. As a result, no additional anonymization or content filtering was required for this work.

G ARTIFACT LICENSING AND INTENDED USE

We use pretrained Gemma and OLMo LMs in accordance with their respective licenses. Gemma models are released under the Apache License 2.0 and subject to Google's use policy, and OLMo models are released under the Apache License 2.0. These licenses permit research use and modification under their stated terms. Our use of these models is consistent with their intended purpose as general-purpose research models, and all artifacts created in this work (e.g., code and experimental configurations) are intended solely for research.

H TRAINING DETAILS

H.1 LORA FINETUNING DETAILS

Table 5: LoRA fine-tuning hyperparameters. The exact number of optimization steps varies for each model based on the size of the training data.

Category	**Hyperparameter**	**Value**
Training schedule	Base learning rate	{3e-5, 5e-5, 1e-4, 3e-4}
	Warmup steps	5
	Schedule	Linear warmup $\rightarrow$ cosine decay
	Epochs	2
	Effective batch size	32
Optimization	Optimizer	AdamW
	(β_1, β_2)	0.9, 0.999
	ϵ	1e-8
	Max grad norm	1.0
Regularization	Weight decay	0.01
	Dropout	0.1
LoRA adapter	Rank (r)	16
	Alpha (α)	32

Table 6: LoRA finetuning results across learning rates. The first row for each model is the unmodified (Original) model.

Model	Learning Rate	**GSym-Test**	MMLU STEM	MMLU Humanities	Trivia QA	Truthful QA
Gemma2 2B	Original	0.411	0.446	0.474	0.409	0.424
	3e-6	0.427	0.446	0.474	0.406	0.423
	5e-6	0.360	0.450	0.472	0.407	0.419
	1e-5	0.342	0.453	0.472	0.412	0.411
	2e-5	0.399	0.459	0.468	0.419	0.403
	3e-5	0.480	0.462	0.467	0.419	0.401
	5e-5	**0.579**	0.451	0.464	0.414	0.404
	1e-4	0.562	0.444	0.467	0.417	0.415
	3e-4	0.578	0.454	0.460	0.410	0.397
Gemma2 9B	Original	0.807	0.609	0.620	0.536	0.536
	3e-6	0.810	0.609	0.620	0.535	0.537
	5e-6	0.808	0.610	0.619	0.536	0.538
	1e-5	0.805	0.610	0.619	0.535	0.535
	2e-5	0.793	0.612	0.621	0.547	0.547
	3e-5	**0.850**	0.614	0.623	0.555	0.535
	5e-5	0.668	0.615	0.623	0.560	0.532
	1e-4	0.609	0.617	0.632	0.581	0.528
	3e-4	0.601	0.611	0.628	0.573	0.523
Olmo 7B	Original	0.739	0.507	0.557	0.622	0.468
	3e-6	0.741	0.507	0.556	0.623	0.467
	5e-6	0.745	0.507	0.557	0.622	0.468
	1e-5	**0.746**	0.504	0.556	0.626	0.466
	2e-5	0.644	0.505	0.557	0.630	0.465
	3e-5	0.660	0.507	0.556	0.630	0.465
	5e-5	0.585	0.509	0.559	0.630	0.463
	1e-4	0.616	0.514	0.559	0.634	0.461
	3e-4	0.442	0.514	0.559	0.638	0.447
Olmo 13B	Original	0.742	0.564	0.601	0.703	0.512
	3e-6	0.754	0.565	0.602	0.705	0.512
	5e-6	0.772	0.565	0.603	0.712	0.512
	1e-5	**0.797**	0.566	0.605	0.720	0.513
	2e-5	0.767	0.567	0.605	0.723	0.509
	3e-5	0.775	0.566	0.605	0.724	0.509
	5e-5	0.776	0.567	0.606	0.726	0.507
	1e-4	0.697	0.569	0.601	0.727	0.504
	3e-4	0.540	0.567	0.605	0.737	0.499

Table 7: Comparison of Model Components Across Models and Conditions as Identified by DCM.

Model Name	**Condition**	**Q Heads**	**K Heads**	**V Heads**	**MLP Neurons**
Gemma-2-9B-It	Branching	337 ± 110	194 ± 24	135 ± 32	649 ± 449
Gemma-2-9B-It	Prefix	239 ± 40	152 ± 10	99 ± 17	372 ± 29
Gemma-2-2B-It	Branching	180 ± 1	75 ± 4	61 ± 4	3969 ± 398
Gemma-2-2B-It	Prefix	119 ± 22	50 ± 7	38 ± 7	1782 ± 329
Olmo-2-1124-7B-It	Branching	376 ± 29	13 ± 1	47 ± 5	494 ± 43
Olmo-2-1124-7B-It	Prefix	250 ± 21	10 ± 1	31 ± 4	429 ± 14
Olmo-2-1124-13B-It	Branching	767 ± 34	13 ± 0	92 ± 3	1567 ± 79
Olmo-2-1124-13B-It	Prefix	667 ± 12	14 ± 1	73 ± 2	1264 ± 60