PARAMETER PATCHING: CAUSAL ANALYSIS OF LEARNED COMPUTATIONS IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models increasingly solve complex reasoning and coding tasks, yet learned computations supporting these capabilities remain poorly understood. Activation-based methods, such as activation patching (AP), reveal how internal states contribute to behavior; however, they leave open how these capabilities arise from the model parameters themselves. To study this dependence directly, we introduce **Parameter Patching (PP)**, which probes components through fixed edits to their parameters. As the parameter counterpart of AP, PP modifies the learned transformation applied throughout execution, rather than substituting the activation states it produces. Across tasks of varying complexity, we find that both methods locate components with causal influence on behavior, yet the components they select serve different behavioral functions. On classical interpretability tasks such as IOI and arithmetic, the two methods indeed select heads with distinct characteristics. For instance, on IOI, PP selects heads that more strongly affect the preference between candidate names than those selected by AP. This divergence matters more on complex reasoning and coding tasks, where PP selections generally incur larger accuracy losses under matched knockouts, indicating that they identify components on which these capabilities more strongly depend. Finally, PP selections also translate into practical gains. Fine-tuning PP-selected heads yields larger improvements than fine-tuning AP-selected ones on both mathematical reasoning and agentic coding tasks, and the same heads support effective activation steering. Together, these results show that intervening on learned transformations reveals behavioral dependencies complementary to activation-based analysis and identifies useful targets for improving model capabilities.

1 INTRODUCTION

Large language models can solve multi-step reasoning and coding tasks (DeepSeek-AI, 2025; Yang et al., 2024), yet the learned computations that support these capabilities remain difficult to characterize (Lindsey et al., 2025). Understanding which components matter, and how they matter, can explain model behavior and identify targets for changing it. Mechanistic interpretability has made progress by tracing internal states to predictions (Wang et al., 2023). A complementary and less explored question is *how the learned transformations encoded in the parameters themselves support these capabilities.*

A mainstream paradigm in mechanistic interpretability focuses on model activations. Activation patching (AP) addresses causal questions by replacing a component's activation in a clean execution with the *corresponding activation from a corrupted one* and measuring the resulting change in behavior (Zhang & Nanda, 2024; Heimersheim & Nanda, 2024). It has revealed mechanisms underlying indirect object identification (IOI), arithmetic and factual recall (Wang et al., 2023; Zhang et al., 2024; Meng et al., 2022). However, an activation is jointly determined by a component's learned transformation and its current input; replacing it therefore conflates the two and cannot by itself distinguish the contribution of the learned transformation from that of the input it happens to receive. This limitation is especially salient when behavior unfolds over multiple generated tokens, where the same transformation is applied repeatedly to changing inputs and its effect compounds over the course of execution.

In this paper, we introduce **Parameter Patching (PP)** to study the contribution of learned transformations directly. As the parameter counterpart of AP, PP applies a fixed edit to a component's parameters

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.

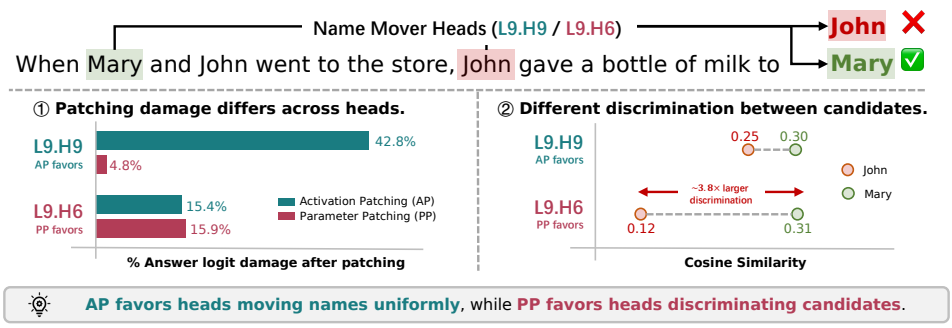


Figure 1: Two GPT-2 Small name mover heads show different profile on displayed example. **Left:** AP strongly favors head L9.H9, which yields low damage after PP. **Right:** Cosine similarities show PP favors heads discriminating candidate names (L9.H6) than heads moving names uniformly (L9.H9).

and reruns the model. The edited transformation acts on the inputs encountered throughout execution, including after earlier changes alter the generated history. *Unlike AP, our procedure requires no paired corrupted input and no alignment of activations across executions, making it directly applicable to complex, free-form generation tasks.* Figure 1 illustrates this distinction using two GPT-2 heads classified as name movers (Wang et al., 2023). On the displayed IOI prompt, PP favors a head whose output more strongly distinguishes the correct name from the competing name, whereas AP favors a head whose output is similarly aligned with both names. This highlights a functional distinction within the name-mover category: promoting candidate names and discriminating which name is correct.

We evaluate PP and AP across tasks of varying complexity to examine what their interventions reveal about model behavior. Both methods locate components with causal influence, but their selections exhibit distinct behavioral characteristics. In our IOI analyses, PP-selected heads more strongly affect the preference between candidate names, whereas AP-selected heads more strongly affect the total probability assigned to those names. Our head analyses further show that PP can identify heads that affect accuracy even when their activation readouts do not favor the correct answer, distinguishing causal importance from answer-token visibility (Section 4.2). On complex mathematical reasoning and coding tasks, PP selections generally cause greater accuracy losses than AP selections under common knockouts, identifying components on which these capabilities depend more strongly. This advantage often persists when selections are transferred across tasks without reselection, demonstrating their relevance beyond the task used for localization (Section 4.3).

Beyond locating influential components, we investigate whether PP selections also make useful targets for model improvement (Section 4.4). We exploit them in two ways. First, at inference time, steering the activations of PP-selected heads improves mathematical reasoning. On GSM8K with Qwen3-4B, PP steering reaches 79.8% accuracy, compared with 71.9% for AP and 68.2% for a layer-matched random control, with gains over baseline and random also on Llama-3-8B. Second, the same selections identify which parameters to train. Fine-tuning only the selected heads while freezing the rest of the model yields higher GSM8K accuracy for PP- than for AP-selected heads. This training advantage extends beyond mathematical reasoning. In a preliminary study on agentic coding tasks SWE-rebench (Badertdinov et al., 2025; 2026), PP-guided training of Qwen3.8-27B reaches 66.7% mean accuracy, outperforming both AP-guided training (63.7%) and full-parameter fine-tuning (64.7%).

Overall, parameter patching complements activation-based analysis by examining how behavior depends on learned transformations reused throughout execution. As language models are increasingly deployed as agents that plan, code, and act over many steps, PP offers a practical handle for both understanding and improving the components behind these capabilities.

2 Preliminaries

Model computation and notation. Let $\mathcal{M}_{\boldsymbol{\theta}}$ denote a language model with fixed computational structure and parameter collection $\boldsymbol{\theta}$. Its modules form a collection $\mathcal{G}$. Each module g specifies a computation and uses the corresponding parameters $\boldsymbol{\theta}_g$. Modules with shared parameters refer to the

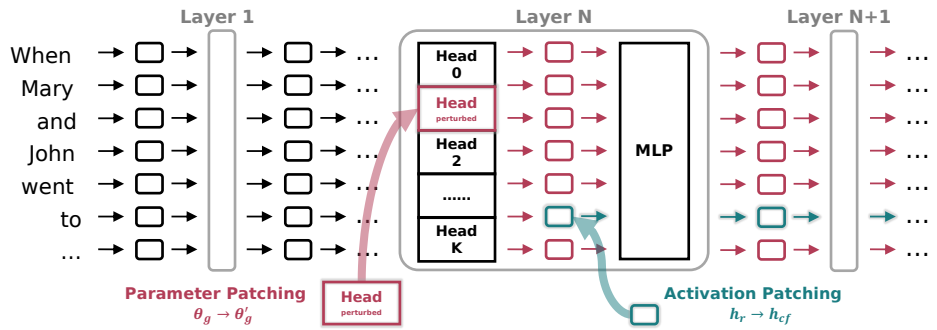


Figure 2: Overview of parameter and activation patching. PP edits a head's parameters, changing the learned transformation reused across positions throughout execution. AP substitutes a corrupted activation at some selected sites in the clean execution. The schematic contrasts an edit to a shared transformation with replacement of a particular activation state.

same entries in $\boldsymbol{\theta}$. At token position t, a module computes

$$\mathbf{a}_{g,t} = g(\mathbf{u}_{g,t}; \boldsymbol{\theta}_g), \tag{1}$$

where $\mathbf{u}_{g,t}$ is its input and $\mathbf{a}_{g,t}$ the corresponding output activation. The computation g and parameters $\boldsymbol{\theta}_g$ are shared across positions, while inputs and activations vary over the execution. We omit the module index when discussing one module and the position index in local calculations.

Activation patching. Activation patching (AP) replaces selected activations in one execution with corresponding activations from another (Vig et al., 2020; Geiger et al., 2021). Let x denote a clean input and x_{cf} its corrupted counterpart. To measure damage, we replace a module's outputs during the clean execution with outputs recorded during the corrupted execution (Wang et al., 2023; Zhang & Nanda, 2024). Let $\mathcal{P}$ be the selected positions in clean execution, $\pi(t)$ the aligned position in corrupted execution, and $\mathbf{a}_{\text{cf},\pi(t)}$ the corresponding output of same module under original parameters. The intervened execution is

$$\mathcal{M}_{\boldsymbol{\theta}}\big(x;\ \mathbf{a}_t \leftarrow \mathbf{a}_{\text{cf},\pi(t)} \text{ for } t \in \mathcal{P}\big)\,. \tag{2}$$

The run on x continues under the specified downstream propagation rule, with model parameters unchanged. The position set may contain one, several or all positions.

Let $\mu(\mathcal{M}, x)$ be a scalar measure of the behavior of interest on input x. We compare its values in the clean and patched executions to quantify the effect of replacing the selected activations. For a higher-is-better metric, damage is the clean value minus the patched value. Comparing damage across modules identifies which activation replacements most disrupt the measured behavior.

3 Parameter Patching

Parameter patching (PP) measures how model behavior depends on a module's learned transformation by changing its parameters and rerunning the model. As the parameter counterpart of AP, it follows the same logic of intervention and behavioral comparison. The edited parameters remain fixed throughout execution, and influence activations across all token positions (Figure 2).

3.1 Fixed interventions on learned transformations

A parameter edit T_g maps the model's parameters to replacement parameters for a module g before execution:

$$\boldsymbol{\theta}'_g = T_g(\boldsymbol{\theta}). \tag{3}$$

Taking the full collection $\boldsymbol{\theta}$ as input allows the replacement to be constructed from the parameters of other modules, for instance by substituting a reference transformation in place of the original one. We write $\boldsymbol{\theta}'$ for the full parameter collection after the edit, with all parameter entries outside the edited block unchanged.

We then run the edited model $\mathcal{M}_{\boldsymbol{\theta}'}$ on the original clean input x, recomputing downstream states normally. At its t-th use during prompt processing and autoregressive generation, the edited module computes

$$\mathbf{a}'_t = g(\mathbf{u}'_t; \boldsymbol{\theta}'_g), \tag{4}$$

where $\mathbf{u}'_t$ and $\mathbf{a}'_t$ are the module's input and output at that use. The parameters $\boldsymbol{\theta}'_g$ remain fixed at every use, and the model's computational structure and parameter sharing are preserved.

Relation to activation interventions. AP supplies replacement activations from a corrupted execution at selected positions, which may also include all positions (Zhang & Nanda, 2024). PP applies a shared parameter edit and computes activations from the inputs at each position. Because these inputs vary, the same edit can produce different activation changes across tokens. The edited transformation therefore remains applicable as the generated context evolves, making PP straightforward to apply throughout autoregressive generation without requiring a paired corrupted input or alignment between executions. Appendix A develops the mathematical relationship between the interventions.

3.2 Measuring and interpreting intervention effects

We compare the original and edited models on the same inputs using the behavioral metric μ defined in Section 2. For a higher-is-better metric, the mean damage over a finite evaluation set $\mathcal{D}$ is

$$\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \left[\mu(\mathcal{M}_{\boldsymbol{\theta}}, x) - \mu(\mathcal{M}_{\boldsymbol{\theta}'}, x) \right]. \tag{5}$$

Positive damage indicates worse performance, while negative damage indicates improvement. When the edit replaces a module with a reference transformation, this comparison measures the model's dependence on the original transformation relative to that reference.

To locate modules supporting the behavior of interest, we edit each module separately in the original model and rank modules by their mean damage. Higher-ranked modules are those whose parameter edits most strongly reduce the chosen metric. In our experiments, we study individual attention heads. We evaluate PP-selected and AP-selected heads on held-out examples using a common knockout, and use the selected heads as targets for steering and training. Section 4.1 specifies the parameter edit and evaluation protocol; Appendix A gives the head transformations.

4 Experiments

We examine what parameter-level interpretation reveals about model behavior and how it can guide targeted improvement. We first compare the behavioral contributions of PP- and AP-selected heads on controlled tasks (Section 4.2), then test their importance for complex capabilities (Section 4.3). Finally, we evaluate these selections as targets for steering and pinpoint training (Section 4.4).

4.1 Experimental Setup

Tasks and models. We compare PP and AP (1) on controlled tasks to characterize the behavior of selected heads, and (2) on mathematical reasoning and code generation to examine their importance for more complex capabilities. The localization and knockout comparisons cover 15 pretrained models from the GPT-2, Llama, Qwen and Gemma families. IOI and arithmetic cover 14 and 13 models, respectively, and require a single-token answer. For complex tasks, we select heads directly on GSM8K (Cobbe et al., 2021) and HumanEval (Chen et al., 2021), and test transfer from arithmetic to GSM8K and from MBPP (Austin et al., 2021) to HumanEval while keeping the source-task selections fixed. Details in Appendices B.1 and D.

Interventions. PP replaces a head's output projection matrix with the mean of the other heads' matrices in its layer, rescaled to the original Frobenius norm. References are fixed from the original model; edits apply at every position with normal downstream recomputation (Appendix A). Following Wang et al. (2023) and Zhang et al. (2024), AP replaces head activations with corrupted-input values at the prediction position while holding later attention outputs fixed. For generation-task localization, this substitution covers continuation-prediction positions, with the same fixed continuation history after both prompts. Corruption protocols appear in Appendices B.1 and B.2.

Family	Size	IOI AP	IOI PP	Arithmetic AP	Arithmetic PP
GPT-2	Small	**.408**	.575	n/a	
Llama	2-7B	.631	**.056**	.394	**.042**
	3.1-8B	.534	**.100**	.906	**.198**
	3.2-1B	.439	**.048**	.550	**.039**
	3.2-3B	.506	**.043**	.641	**.095**
Qwen3	0.6B	.500	**.031**	.359	**.032**
	1.7B	.581	**.346**	.626	**.035**
	4B	**.580**	.657	.568	**.383**
	8B	**.591**	.740	.719	**.594**
Qwen3.5	0.8B	**.198**	.419	.375	**.125**
	2B	**.162**	.179	.837	**.300**
	4B	**.373**	.799	.903	**.786**
Gemma4	E2B	**.118**	.192	.857	**.706**
	E4B	**.140**	.281	**.453**	.664

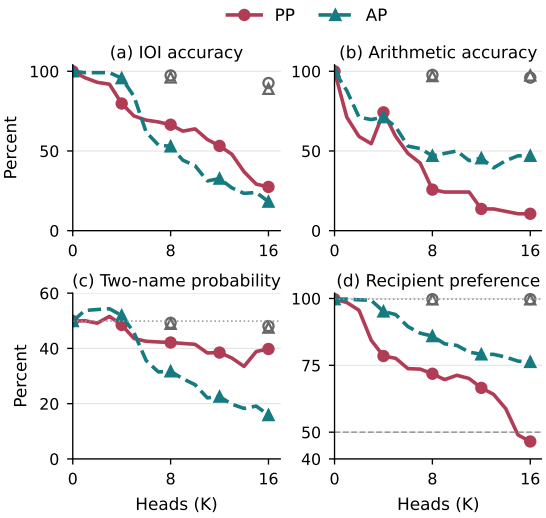


Table 1: Knockout effects of PP- and AP-selected heads on IOI and arithmetic. **Left:** normalized retained-accuracy AUC over $K = 0$–16; bold shading marks greater damage. **Right:** Qwen3-4B accuracy (a,b), total candidate-name probability (c), and conditional correct-name probability (d). Gray hollow markers denote random controls; dotted lines show intact probabilities and the dashed line equal name preference. Full curves in Appendices C.2 and C.3.

Head selection and metrics. We rank heads independently for PP and AP by mean signed damage (Equation (5)). The metric μ is recipient-minus-subject logit margin for IOI, correct-answer probability for arithmetic, and mean token log-probability of model's own correct continuation for GSM8K, HumanEval and MBPP under teacher-forcing. Selection and knockout use disjoint, fixed sets of initially correct examples; generation correctness uses completed greedy decoding.

Evaluation. We assess head selections through knockout, activation steering and pinpoint training (Chen et al., 2024). Knockout measures behavioral dependence using a common mean-activation replacement across prompt and generated positions. Steering modifies selected activations to test inference-time improvement; pinpoint training updates selected parameters while freezing the rest to test targeted adaptation. Layer-matched random heads and full-model fine-tuning provide application-specific controls. We measure answer accuracy on controlled and mathematical tasks, original-test functional pass@1 on HumanEval, and task success on SWE-rebench. Generation outcomes are scored from newly generated outputs. Controlled-task secondary measures include target-token negative log-likelihood (NLL) and candidate-name probabilities. We summarize knockout curves by the area under the retained-accuracy curve (AUC) over 0–16 heads, computed by trapezoidal integration and divided by 16; lower AUC means greater disruption. More details appear in the corresponding subsections and appendices (Appendices B.2, C.2 and E).

4.2 From Head Selection to Behavioral Characterization

We begin with IOI and arithmetic, where the restricted answer spaces make the behavioral effects of head interventions easier to characterize. Using the common knockout protocol, we first test whether PP and AP select heads that influence task performance. We then examine answer probabilities and individual-head readouts to understand how their contributions differ.

Both methods find heads the model relies on. PP- and AP-selected heads both cause greater accuracy loss than layer-matched random heads in the 6 controlled comparisons (Appendix C.2). We examine answer behavior at $K = 8$ and summarize retained accuracy over $K = 0$–16, focusing on the substantial effects of small head sets (Table 1). Yet the methods select different components: the Qwen3-4B arithmetic selections share only 1 of their top 8 heads. We therefore examine which aspects of answer behavior these different selections affect.

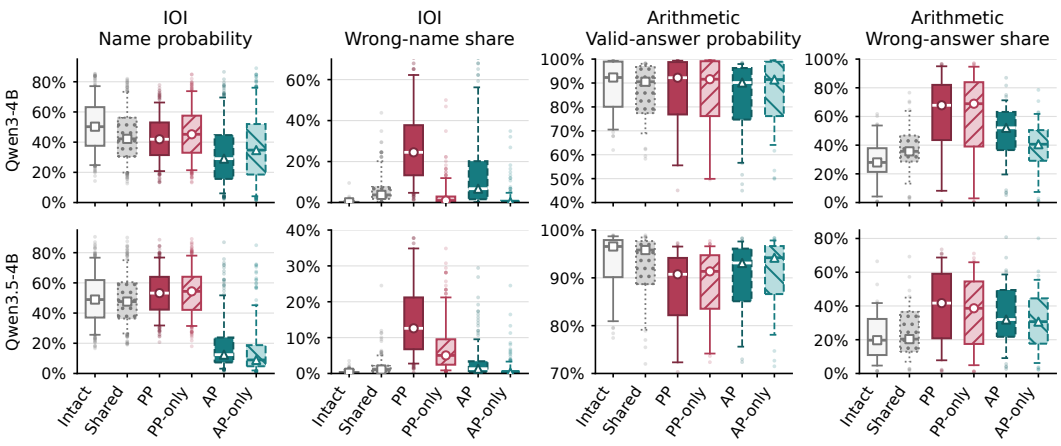


Figure 3: Candidate probability and answer preference under head knockouts on IOI and arithmetic. Rows show Qwen3-4B and Qwen3.5-4B; groups compare intact, PP-selected, AP-selected, shared and exclusive heads. Further analysis in Appendix C.3.

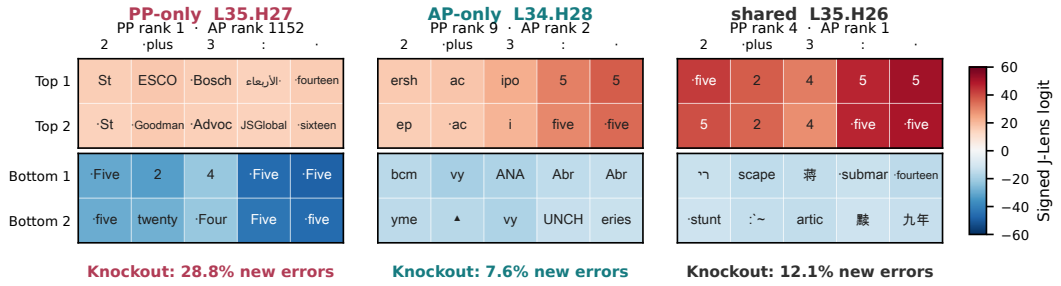


Figure 4: Arithmetic head readouts and knockout effects in Qwen3-4B for $2 + 3$. Columns show PP-only, AP-only and shared heads; cells list the two highest- and lowest-scoring tokens under J-Lens. Red and blue denote positive and negative logits. Knockout rates measure new errors across the initially correct arithmetic evaluation set.

PP and AP highlight different behavioral contributions. In Qwen3-4B, PP-selected knockouts more strongly disrupt discrimination among candidate answers than AP-selected knockouts. On IOI, AP causes greater accuracy loss and a larger reduction in total probability on the two candidate names, whereas PP shifts a larger share of that probability toward the incorrect name (Figure 3). On arithmetic, both knockouts leave similar total probability on valid answers, but PP assigns a larger share to incorrect sums. PP-only heads largely reproduce full PP selection's effect on sum preference, linking this distinction to heads not selected by AP. More comparisons are in Appendix C.3.

Head readouts complement knockout effects. In Qwen3-4B, PP identifies additional influential heads whose vocabulary readouts do not prominently favor the correct answer. The Jacobian lens (J-Lens; Gurnee et al., 2026) maps head outputs to vocabulary scores, which we examine alongside individual-head knockout effects. In the IOI example (Appendix C.4), the AP-only head more clearly favors the correct recipient in its readout, but knocking out the PP-only head causes more errors across the evaluation set. In the arithmetic example (Figure 4), the PP-only head gives the correct sum a negative readout score. Its knockout causes the largest accuracy loss among the three displayed heads, with errors that copy an operand.

4.3 From Controlled Tasks to Complex Capabilities

These controlled comparisons distinguish different contributions to answer behavior. We next ask whether PP identifies heads important for successful reasoning and code generation, and whether their influence extends beyond the task used to select them.

Family	Size	GSM8K AP	GSM8K PP	HumanEval AP	HumanEval PP
Llama	2-7B	.563	**.384**	.391	**.031**
	3-8B	.739	**.509**	.829	**.357**
	3.1-8B	.743	**.539**	.623	**.508**
	3.2-1B	.230	**.106**	.516	**.297**
	3.2-3B	.545	**.164**	.609	**.258**
Qwen3	0.6B	.746	**.035**	.234	**.028**
	1.7B	.851	**.033**	.777	**.030**
	4B	.700	**.522**	.851	**.738**
	8B	.779	**.696**	.866	**.605**
Qwen3.5	0.8B	.256	**.054**	.506	**.050**
	2B	.221	**.144**	.467	**.100**
	4B	**.854**	.915	.707	**.558**
Gemma4	E2B	.150	**.057**	**.120**	.125
	E4B	.265	**.212**	.341	**.072**

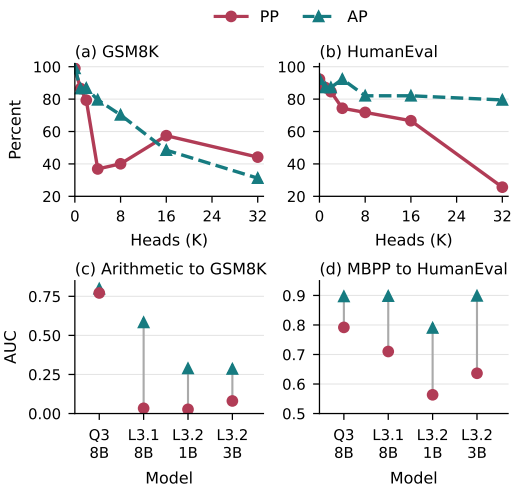


Table 2: Direct localization and transfer on GSM8K and HumanEval. **Left:** normalized retained-accuracy AUC over $K = 0$–16; bold shading marks greater damage. **Right:** Qwen3-4B accuracy through $K = 32$ (a,b; axes start at 20%), and paired transfer AUCs from arithmetic to GSM8K (c) and MBPP to HumanEval (d). Full results in Appendix D.

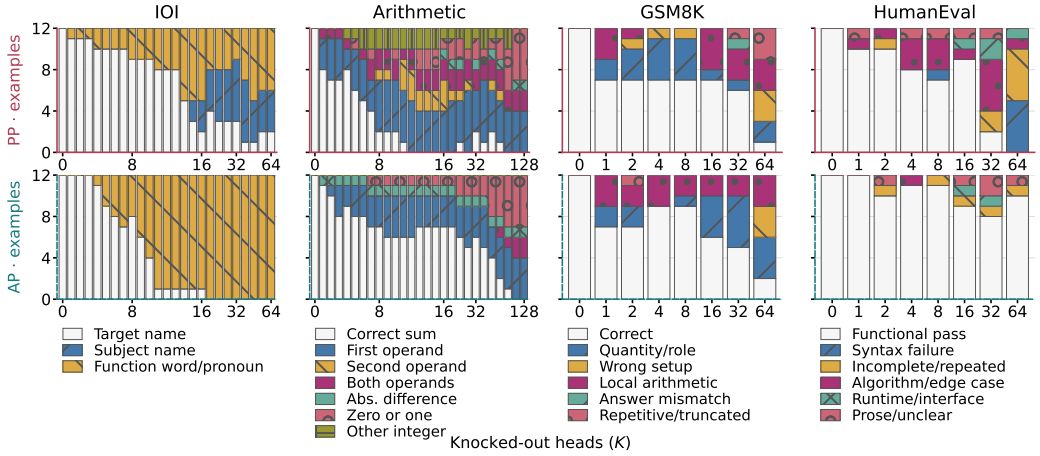


Figure 5: Output categories under Qwen3-4B head knockouts, with PP above AP. Each task follows 12 initially correct examples. Head counts use equal spacing despite unequal increments. Annotation details are in Appendix D.3.

Direct localization in generation. Both AP and PP locate heads whose influence extends beyond the fixed continuations used for selection to the correctness of newly generated solutions. On GSM8K and HumanEval, the common knockout generally causes greater accuracy loss for PP-selected heads than for AP-selected heads, as reflected in lower retained-accuracy AUCs over $K = 0$–16 (Table 2). The Qwen3-4B GSM8K rebound also shows that increasing the number of selected heads need not increase damage (Appendix D.3).

Cross-task transfer. Both AP and PP identify heads whose influence extends to related tasks. Keeping each method's source-task rankings fixed, we evaluate arithmetic-selected heads on GSM8K and MBPP-selected heads on HumanEval. Under the common knockout, PP selections yield lower retained-accuracy AUCs than AP selections in most arithmetic-to-GSM8K comparisons and all MBPP-to-HumanEval comparisons over $K = 0$–16 (Tables 2 and 10).

			Steering				Training			
Family	**Size**	**Base**	**Random**	**AP**	**PP**	**Δ**	**Random**	**AP**	**PP**	**Δ**
Llama	2-7B	3.9	4.0	4.3	**5.2**	+0.9	13.9	13.6	**20.7**	+7.1
	3-8B	22.7	19.9	24.1	**26.5**	+2.4	45.6	45.2	**55.0**	+9.8
	3.1-8B	26.6	—	—	—	—	48.7	45.6	**54.4**	+8.9
	3.2-1B	3.6	3.6	3.6	**3.6**	+0.1	6.1	7.4	**11.7**	+4.3
	3.2-3B	7.6	**8.0**	7.7	7.6	-0.1	24.9	26.0	**34.0**	+8.0
Qwen3	0.6B	54.3	**54.8**	**54.8**	54.7	-0.1	51.6	51.5	**52.5**	+1.0
	1.7B	65.0	62.7	65.2	**65.7**	+0.5	68.2	**69.9**	69.3	-0.6
	4B	70.3	68.2	71.9	**79.8**	+7.9	81.9	81.3	**82.5**	+1.2
	8B	79.3	**80.1**	79.2	79.2	0.0	82.9	83.2	**83.9**	+0.7
Qwen3.5	0.8B	39.4	37.9	38.2	**39.4**	+1.2	44.9	45.6	**48.3**	+2.7
	2B	60.3	—	—	—	—	61.8	60.8	**64.5**	+3.7
	4B	82.9	**83.2**	82.6	82.9	+0.4	80.5	79.6	**85.0**	+5.4
Gemma4	E2B	17.4	17.7	**18.0**	17.7	-0.4	23.0	23.1	**24.6**	+1.4
	E4B	37.4	40.3	**41.7**	40.0	-1.7	55.3	55.7	**59.7**	+3.9

Table 3: Zero-shot GSM8K accuracy (%). Base is the lower recorded unmodified repeat. Random denotes the layer-matched random control. Bold shading marks the highest accuracy in each intervention group. Δ is PP minus AP in percentage points, computed before rounding. Dashes indicate no steering selected in calibration. Protocols and controls are in Appendix E.

How the selected heads affect generation. The Qwen3-4B examples illustrate how AP- and PP-selected knockouts affect answer choice and output structure (Figure 5). The IOI and arithmetic outputs reflect the candidate-answer differences above. On GSM8K, both selections lead to calculation errors or incorrect use of the given quantities. On HumanEval, PP-selected knockouts produce invalid or incomplete programs at $K = 64$, while AP retains more correct programs. This difference also appears in complete-cohort MBPP-to-HumanEval transfer (Appendix D.3): at $K = 16$, PP-selected knockouts produce unparseable programs in 58.4% of Qwen3-4B cases, compared with 6.7% for AP.

4.4 From Causal Influence to Targeted Improvement

The knockout studies establish the influence of AP- and PP-selected heads on task performance. We next compare their utility as targets for improvement (Hase et al., 2023), through inference-time activation steering and pinpoint training of the selected parameters.

Inference-time steering. We add directions defined by activation differences between eight-shot and zero-shot prompting. On Qwen3-4B, PP steering reaches 79.8% GSM8K accuracy, compared with 71.9% for AP and 68.2% for random (Table 3). PP also improves Llama-3-8B over its baseline and layer-matched random control; its difference from AP remains unresolved (Appendix E).

Pinpoint training. Following localization-guided training (Zhang et al., 2024; Chen et al., 2024), we update the parameters of selected heads while freezing the rest of the model. On the official GSM8K test, PP-selected-head training achieves higher accuracy than AP-selected-head training in 13 of 14 models and the layer-matched random control in all 14 (Table 3). The advantage spans Llama, Qwen and Gemma checkpoints: PP reaches 55.0% versus AP's 45.2% on Llama-3-8B, 85.0% versus 79.6% on Qwen3.5-4B, and 59.7% versus 55.7% on Gemma4-E4B. Full supervised fine-tuning (SFT) and training costs appear in Appendix E. These results support PP as a criterion for selecting effective adaptation targets. PP measures how changing a learned transformation affects task performance, including its downstream consequences; pinpoint training updates those same

Condition	**Accuracy (%)**
Unmodified	64.0
Full SFT	64.7
AP heads	63.7
PP heads	**66.7**

Table 4: SWE-rebench accuracy (%) for Qwen3.8-27B. Bold shading marks the highest accuracy.

transformations. This alignment between selection and adaptation offers an explanation for the observed gains.

Software-task training. A preliminary SWE-rebench study (Badertdinov et al., 2025) extends the AP–PP training comparison to software-agent adaptation. We fine-tune Qwen3.8-27B on a sampled training subset using trajectories generated by Qwen3.8-Max, then evaluate on a disjoint, held-out subset from the same domain. Across 100 test problems with 3 sampled generations per problem, PP-selected-head training achieves 66.7% mean accuracy, compared with 63.7% for AP-selected-head training, 64.7% for full SFT and 64.0% for the original model (Table 4). This result supports using PP to select training targets for software-agent adaptation.

5 Related Work

We relate PP to activation-based and parameter-based interpretability, and to the use of localized components for targeted model improvement.

Activation-based interpretability. Activation and path patching connect internal states to model behavior (Wang et al., 2023; Zhang & Nanda, 2024; Heimersheim & Nanda, 2024). Automated circuit discovery searches for relevant connections (Conmy et al., 2023), while attribution patching approximates intervention effects with gradients (Syed et al., 2023; Hanna et al., 2024; Kramár et al., 2024). Learned head gates also characterize functional contributions (Nam et al., 2025). Sparse autoencoders decompose activations into learned features (Bricken et al., 2023; Huben et al., 2024); sparse feature circuits and transcoders trace interactions among learned features (Marks et al., 2025; Dunefsky et al., 2024; Ameisen et al., 2025). These approaches combine the choice of explanatory units with analysis of their contribution to computation.

Parameter-based interpretability. Learned transformations can also be studied through their parameters. Vocabulary-space analyses infer candidate functions from model weights (Dar et al., 2023; Elhelo & Geva, 2025). Parameter decompositions learn additive components of model weights, providing a counterpart to activation feature decomposition (Braun et al., 2025; Bushnaq et al., 2025). Differentiable weight masks identify functional subnetworks (Csordás et al., 2021), while Weight Patching transfers weights from a specialized checkpoint to study recovery of an internal control state (Sun et al., 2026). Our PP instantiation measures how task behavior depends on existing head transformations using fixed references constructed within the original checkpoint, and compares the resulting selections with those of AP.

Localization-guided improvement. Localized components also provide targets for constructive interventions. ROME and MEMIT connect causal tracing to factual weight editing (Meng et al., 2022; 2023). Supervised pinpoint tuning and arithmetic-specific tuning update behavior-relevant modules (Chen et al., 2024; Zhang et al., 2024). Function vectors extracted from causally identified heads elicit task behavior (Todd et al., 2024); inference-time intervention shifts selected head activations (Li et al., 2023), while LoFiT learns offsets at selected head representations (Yin et al., 2024). Causal-tracing scores need not predict factual-editing success (Hase et al., 2023), motivating our separate evaluation of AP- and PP-selected heads for steering and pinpoint training.

6 Conclusion

Parameter patching provides a direct way to connect learned transformations to model behavior. As the parameter counterpart of activation patching, it tests the consequence of changing a component's learned mapping while allowing the model to compute the resulting states throughout execution. This connects mechanistic analysis to the computations encoded in model parameters, complementing explanations based on activation states. The significance of this perspective lies in linking two questions: how a model's capabilities depend on its learned components, and where interventions can change those capabilities. Our study shows how parameter interventions can make that connection useful for both behavioral interpretation and targeted improvement. PP thus offers a practical basis for studying learned computation across inputs and generated sequences, bringing causal understanding closer to deliberate changes in model behavior.

Reproducibility Statement

Section 3 and Appendix A define the interventions and their first-order relations. Section 4.1 and Appendices B and E describe task construction, model identities, eligibility, scoring and constructive evaluations. The appendices report full model comparisons, uncertainty, numerical diagnostics and training protocols. Code will be made available after publication.

Ethics Statement

The study evaluates language-model components on controlled prompts, mathematical reasoning and code benchmarks. Targeted interventions can improve capabilities and have dual-use implications. Benchmark performance does not establish deployment safety or the faithfulness of generated reasoning. The reported interventions should be evaluated for their intended use and effects on other capabilities before deployment.

AI Use Statement

We used generative AI tools to aid and polish writing, for literature retrieval and discovery, and for research ideation and execution. Writing assistance covered language, organization, notation and clarity; retrieval assistance helped identify relevant literature and check bibliographic information. Research assistance included methodological reasoning, experiment planning, code development, and analysis and interpretation of results, including the qualitative output annotation described in Appendix D.3. We reviewed the AI-assisted manuscript and checked reported comparisons against the study records. Implementation checks included numerical tests and experiment audits. The authors take responsibility for the final text, scientific claims, references and research artifacts, including material produced with AI assistance.

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.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. Swe-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/21bec6ace947b1b58967b945c8ac0f10-Abstract-Datasets_and_Benchmarks_Track.html.

Ibragim Badertdinov, Maksim Nekrashevich, Anton Shevtsov, and Alexander Golubev. Swe-rebench V2: language-agnostic SWE task collection at scale. *CoRR*, abs/2602.23866, 2026. doi: 10.48550/ARXIV.2602.23866. URL https://doi.org/10.48550/arXiv.2602.23866.

Dan Braun, Lucius Bushnaq, Stefan Heimersheim, Jake Mendel, and Lee Sharkey. Interpretability in parameter space: Minimizing mechanistic description length with attribution-based parameter decomposition, 2025. URL https://doi.org/10.48550/arXiv.2501.14926.

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

Lucius Bushnaq, Dan Braun, and Lee Sharkey. Stochastic parameter decomposition. *CoRR*, abs/2506.20790, 2025. doi: 10.48550/ARXIV.2506.20790. URL https://doi.org/10.48550/arXiv.2506.20790.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Wei Chen, Zhen Huang, Liang Xie, Binbin Lin, Houqiang Li, Le Lu, Xinmei Tian, Deng Cai, Yonggang Zhang, Wenxiao Wang, Xu Shen, and Jieping Ye. From yes-men to truth-tellers: Addressing sycophancy in large language models with pinpoint tuning. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6950–6972. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/chen24u.html.

Xi Chen, Mingyu Jin, Jingcheng Niu, Yutong Yin, Jinman Zhao, Bangwei Guo, Dimitris N. Metaxas, Zhaoran Wang, Yutao Yue, and Gerald Penn. All circuits lead to rome: Rethinking functional anisotropy in circuit and sheaf discovery for llms. *CoRR*, abs/2605.12671, 2026. doi: 10.48550/ARXIV.2605.12671. URL https://doi.org/10.48550/arXiv.2605.12671.

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.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Abstract-Conference.html.

Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Are neural nets modular? inspecting functional modularity through differentiable weight masks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=7uVcpu-gMD.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. Analyzing transformers in embedding space. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the*

61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023, pp. 16124–16170. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.893. URL https://doi.org/10.18653/v1/2023.acl-long.893.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://doi.org/10.48550/arXiv.2501.12948.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/2b8f4db0464cc5b6e9d5e6bea4b9f308-Abstract-Conference.html.

Amit Elhelo and Mor Geva. Inferring functionality of attention heads from their parameters. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 17701–17733. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.866. URL https://doi.org/10.18653/v1/2025.acl-long.866.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 9574–9586, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/4f5c422f4d49a5a807eda27434231040-Abstract.html.

Wes Gurnee, Nicholas Sofroniew, Adam Pearce, Mateusz Piotrowski, Isaac Kauvar, Runjin Chen, Anna Soligo, Paul C. Bogdan, Euan Ong, Rowan Wang, Ben Thompson, David Abrahams, Subhash Kantamneni, Emmanuel Ameisen, Joshua Batson, and Jack Lindsey. Verbalizable representations form a global workspace in language models. *CoRR*, abs/2607.15495, 2026. doi: 10.48550/ARXIV.2607.15495. URL https://doi.org/10.48550/arXiv.2607.15495.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *CoRR*, abs/2403.17806, 2024. doi: 10.48550/ARXIV.2403.17806. URL https://doi.org/10.48550/arXiv.2403.17806.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/3927bbdcf0e8d1fa8aa23c26f358a281-Abstract-Conference.html.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *CoRR*, abs/2404.15255, 2024. doi: 10.48550/ARXIV.2404.15255. URL https://doi.org/10.48550/arXiv.2404.15255.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. Atp*: An efficient and scalable method for localizing LLM behaviour to components. *CoRR*, abs/2403.00745, 2024. doi: 10.48550/ARXIV.2403.00745. URL https://doi.org/10.48550/arXiv.2403.00745.

Kenneth Li, Oam Patel, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/81b8390039b7302c909cb769f8b6cd93-Abstract-Conference.html.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, 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. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=I4e82CIDxv.

Riccardo Massidda, Atticus Geiger, Thomas Icard, and Davide Bacciu. Causal abstraction with soft interventions. In Mihaela van der Schaar, Cheng Zhang, and Dominik Janzing (eds.), *Conference on Causal Learning and Reasoning, CLeaR 2023, 11-14 April 2023, Amazon Development Center, Tübingen, Germany, April 11-14, 2023*, volume 213 of *Proceedings of Machine Learning Research*, pp. 68–87. PMLR, 2023. URL https://proceedings.mlr.press/v213/massidda23a.html.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Andrew Nam, Henry Conklin, Yukang Yang, Tom Griffiths, Jonathan D. Cohen, and Sarah-Jane Leslie. Causal head gating: A framework for interpreting roles of attention heads in transformers. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/a7f530e11fa19e9551b7a51dbd0f336f-Abstract-Conference.html.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. 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), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 15504–15522. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.828. URL https://doi.org/10.18653/v1/2024.acl-long.828.

Chenghao Sun, Chengsheng Zhang, Guanzheng Qin, Rui Dai, and Xinmei Tian. Weight patching: Toward source-level mechanistic localization in llms, 2026. URL https://doi.org/10.48550/arXiv.2604.13694.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. *CoRR*, abs/2310.10348, 2023. doi: 10.48550/ARXIV.2310.10348. URL https://doi.org/10.48550/arXiv.2310.10348.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. Investigating gender bias in language models using causal mediation analysis. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/92650b2e92217715fe312e6fa7b90d82-Abstract.html.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Fangcong Yin, Xi Ye, and Greg Durrett. Lofit: Localized fine-tuning on LLM representations. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/122ea6470232ee5e79a2649243348005-Abstract-Conference.html.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Hf17y6u9BC.

Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu-ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. Interpreting and improving large language models in arithmetic calculation. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 59932–59950. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/zhang24bk.html.

A Parameter Patching: Implementation and Analysis

A.1 Attention-head intervention and reference construction

For query head h in layer ℓ, the output-write matrix $\mathbf{W}_{O,\ell h}$ maps the attention-aggregated value vector $\mathbf{u}_{\ell h,t}$ to its residual-stream contribution: $\mathbf{a}_{\ell h,t} = g(\mathbf{u}_{\ell h,t}; \mathbf{W}_{O,\ell h}) = \mathbf{W}_{O,\ell h}\mathbf{u}_{\ell h,t}$. Shared output bias is excluded. The same matrix acts at every position; PP interpolates it toward a reference constructed from the other heads:

$$\begin{aligned} \mathbf{C}_{\ell h} &= \frac{1}{H_\ell - 1} \sum_{j \neq h} \mathbf{W}_{O,\ell j}, \\ \mathbf{W}'_{O,\ell h} &= (1-\lambda)\mathbf{W}_{O,\ell h} + \lambda \mathbf{R}_{\ell h}. \end{aligned} \tag{6}$$

Here $H_\ell > 1$ is the number of query heads in layer ℓ and $\lambda \in [0, 1]$ controls strength: zero retains the original matrix and one replaces it completely. The mean averages corresponding weight coordinates across heads; these need not represent aligned semantic functions. The norm-matched reference is $\mathbf{R}_{\ell h} = \mathbf{C}_{\ell h}\|\mathbf{W}_{O,\ell h}\|_F / \|\mathbf{C}_{\ell h}\|_F$. It retains the original matrix's Frobenius norm at full replacement, but does not preserve activation magnitude. Intermediate strengths need not preserve either norm.

Unrescaled means use $\mathbf{R}_{\ell h} = \mathbf{C}_{\ell h}$, and zeroing uses $\mathbf{R}_{\ell h} = \mathbf{0}$. References are computed from the original checkpoint before any intervention and remain fixed across inputs, joint edits and differentiation. Norm matching rejects a zero or near-zero reference when the target matrix is nonzero; a zero target matrix remains zero. An unrescaled mean may be zero.

Each intervention acts on distinct stored parameter entries, even when those entries are used at several positions. Qwen3.5 additionally has recurrent value-head output units, and Gemma4 has query heads of different widths. Their replacements average compatible coordinates within a layer. Equal head counts therefore need not give equal parameter counts across architectures or selected sets.

A.2 Input-dependent activation changes

PP replaces the module's parameterized input–output map while preserving its computational structure, a soft intervention (Massidda et al., 2023). Using the positions $\mathcal{P}$ and alignment π from Equation (2), the corrupted execution supplies $\mathbf{a}_{\mathrm{cf},\pi(t)} = g(\mathbf{u}_{\mathrm{cf},\pi(t)}; \boldsymbol{\theta}_g)$. The interventions are

$$\begin{aligned} \mathbf{a}_t^{\mathrm{AP}} &= \mathbf{a}_{\mathrm{cf},\pi(t)}, \qquad t \in \mathcal{P}, \\ \mathbf{a}'_t &= g(\mathbf{u}'_t; \boldsymbol{\theta}'_g), \quad \text{every use of } \boldsymbol{\theta}'_g. \end{aligned} \tag{7}$$

AP supplies the recorded corrupted activation at each selected site, which may include all positions. PP recomputes the output from the current input using the same edited parameters at every invocation.

For a bias-free linear module $\mathbf{a} = \mathbf{W}\mathbf{u}$, the local changes from the same baseline output are

$$\delta\mathbf{a}_{\mathrm{AP}} = \mathbf{W}(\mathbf{u}_{\mathrm{cf}} - \mathbf{u}), \qquad \delta\mathbf{a}_{\mathrm{PP}} = (\mathbf{W}' - \mathbf{W})\mathbf{u}. \tag{8}$$

Here $\mathbf{u}_{\mathrm{cf}}$ is the aligned corrupted input at one invocation, and both PP matrices are evaluated at the same $\mathbf{u}$. The shared edit can thus produce different local activation changes for different inputs.

Dependence on the current input. For a fixed corrupted output $\mathbf{a}_{\mathrm{cf}}$, full AP replacement gives $\mathbf{a}^{\mathrm{AP}}(\mathbf{u}) = \mathbf{a}_{\mathrm{cf}}$, whereas $\mathbf{a}^{\mathrm{PP}}(\mathbf{u}) = g(\mathbf{u}; \boldsymbol{\theta}'_g)$. Wherever derivatives exist,

$$D_{\mathbf{u}}\mathbf{a}^{\mathrm{AP}} = \mathbf{0}, \qquad D_{\mathbf{u}}\mathbf{a}^{\mathrm{PP}} = D_{\mathbf{u}}g(\mathbf{u}; \boldsymbol{\theta}'_g). \tag{9}$$

The first derivative holds the corrupted output fixed while varying the clean module input. It concerns the fully replaced output, not residual bypasses or the whole model. The PP derivative may also vanish, as with output-weight zeroing.

Changed inputs during execution. When the module input changes from $\mathbf{u}_t$ to $\mathbf{u}'_t$, adding and subtracting $g(\mathbf{u}_t; \boldsymbol{\theta}'_g)$ gives

$$\begin{aligned} \mathbf{a}'_t - \mathbf{a}_t = &[g(\mathbf{u}_t; \boldsymbol{\theta}'_g) - g(\mathbf{u}_t; \boldsymbol{\theta}_g)] \\ &+ [g(\mathbf{u}'_t; \boldsymbol{\theta}'_g) - g(\mathbf{u}_t; \boldsymbol{\theta}'_g)]. \end{aligned} \tag{10}$$

The terms separate the transformation change at the baseline input from the input change passed through the edited transformation; they are not independent causal effects. For a linear module, with $\delta\mathbf{W} = \mathbf{W}' - \mathbf{W}$ and $\delta\mathbf{u}_t = \mathbf{u}'_t - \mathbf{u}_t$, the full change is $\delta\mathbf{W}\mathbf{u}_t + \mathbf{W}\delta\mathbf{u}_t + \delta\mathbf{W}\delta\mathbf{u}_t$. The changed-input terms vanish for a singly edited feedforward output projection on a fixed sequence. Joint edits, recurrent computation or changed generated histories can change the input.

Equivalent specifications. For a bias-free linear contribution, setting $\mathbf{W}' = \mathbf{0}$ is equivalent to zeroing its output, and setting $\mathbf{W}' = c\mathbf{W}$ is equivalent to scaling the output computed from the current input by c. More generally, an activation intervention that evaluates $g(\mathbf{u}_{\text{current}}; \boldsymbol{\theta}'_g)$ can implement the edited function. Exact equality of the full executions requires the same initial state, coverage of every use and downstream recomputation. A fixed corrupted activation need not reproduce that function across inputs. These algebraic identities do not establish finite-precision equality or equate the differently propagated discovery procedures used here.

A.3 First-order relation to activation patching

The experiments rank exact intervention effects. Their first-order companions relate local parameter and activation changes as follows. Identify the parameter collection with a vector containing each distinct scalar coordinate once. For a fixed input and token history, let μ be a differentiable, higher-is-better output metric. With fixed displacement $\delta\boldsymbol{\theta}_g = \boldsymbol{\theta}'_g - \boldsymbol{\theta}_g$, parameter attribution patching (PAtP) predicts damage as

$$\widehat{d}_g^{\,\mathrm{PAtP}} = -\langle \nabla_{\boldsymbol{\theta}_g}\mu, \delta\boldsymbol{\theta}_g \rangle. \tag{11}$$

The gradient is evaluated at the original clean execution and includes every use of the edited parameters. The reference is held fixed; the derivative does not pass through its construction. For zeroing, $\delta\boldsymbol{\theta}_g = -\boldsymbol{\theta}_g$, giving $\widehat{d}_g^{\,\mathrm{PAtP}} = \langle \nabla_{\boldsymbol{\theta}_g}\mu, \boldsymbol{\theta}_g \rangle$.

For AP at positions $\mathcal{P}$, define $\delta\mathbf{a}_t = \mathbf{a}_{\mathrm{cf},\pi(t)} - \mathbf{a}_t$. Activation attribution patching (AtP) (Kramár et al., 2024) predicts damage as

$$\widehat{d}_g^{\,\mathrm{AtP}} = -\sum_{t\in\mathcal{P}} \langle \nabla_{\mathbf{a}_t}\mu, \delta\mathbf{a}_t \rangle. \tag{12}$$

The gradients are evaluated at the original clean execution, with corrupted activations held fixed and the same metric reduction as the exact comparison. Positive predicted damage means a decrease in μ.

Shared parameters and local activation changes. On the same propagation graph, let $\mathbf{J}_t$ differentiate $g(\mathbf{u}_t; \boldsymbol{\theta}_g)$ with respect to $\boldsymbol{\theta}_g$, holding $\mathbf{u}_t$ fixed. The local first-order change is $\mathbf{J}_t\delta\boldsymbol{\theta}_g$. When the invocations include every use of these parameters, the chain rule gives

$$\nabla_{\boldsymbol{\theta}_g}\mu = \sum_t \mathbf{J}_t^\top \nabla_{\mathbf{a}_t}\mu. \tag{13}$$

Activation gradients already include effects propagated to later inputs. Adding those input changes again would double-count the same paths. On this shared graph, matching AP displacements to $\mathbf{J}_t\delta\boldsymbol{\theta}_g$ at every affected use suffices to match first-order metric effects; equal sums of gradient projections suffice for equal scalar predictions.

Each gradient must use its exact comparator's propagation rule. The primary PP and AP procedures use full recomputation and fixed later attention outputs, respectively, so Equation (13) does not equate their predictions. First-order accuracy requires local smoothness and sufficiently small edits. These fixed-history derivatives do not differentiate through discrete token selection or predict generation accuracy. Signed damage predictions retain their sign rather than being replaced by absolute scores.

B Experimental Protocols

B.1 Models, tasks and evaluation populations

Task-specific model coverage appears in Tables 1, 9 and 10. The controlled-task cohorts detailed here use offline snapshots of `openai-community/gpt2`, `meta-llama/Llama-2-7b-hf`, `Qwen/Qwen3-4B-Base` and `Qwen/Qwen3-8B-Base`. Table 5 gives these cohorts, including

Llama-2 IOI. The other controlled-task comparisons use the same candidate splits with model-specific eligibility.

IOI uses 512 discovery and 512 verification candidates per model, sampled from seven ABBA and seven BABA templates, without random prefixes. Clean and corrupted prompts require aligned tokenization and complete single-token targets. For GPT-2 and Qwen3-4B/8B, all 1,024 candidates are token-compatible. Name-pair families do not overlap between discovery and evaluation. ABC corruption substitutes three distinct unrelated names; unlike the role-swap illustration, it need not define an alternative answer.

Arithmetic uses the fixed released data seed 0. Its 180 candidates split into 99 discovery and 81 verification examples with disjoint unordered-operand families. All candidates are token-compatible. Eligibility keeps baseline-correct examples without refilling either split.

The arithmetic task evaluates $A + B$ for $A = 1, \ldots, 5$ and $B = 1, \ldots, 4$, expressed through nine templates. Its paired corruption removes the operation using the released `void`/`empty`/`nothing` construction. This corruption also need not have a correct alternative answer.

Model	Task	Discovery	Evaluation	Families	Precision
GPT-2 Small	IOI	458	454	428	FP32
Llama-2-7B	IOI	5	5	5	BF16
Qwen3-4B	IOI	324	346	331	BF16
Qwen3-8B	IOI	342	364	350	BF16
Llama-2-7B	Arithmetic	99	77	7	FP32
Qwen3-4B	Arithmetic	86	66	7	BF16
Qwen3-8B	Arithmetic	98	75	7	BF16

Table 5: Controlled-task cohorts for GPT-2, Llama-2 and Qwen3, after baseline-correct screening. Families count evaluation name-pair or unordered-operand clusters. Candidate splits precede model-specific correctness screening.

GPT-2 IOI and Llama-2 arithmetic eligibility use Transformers full-vocabulary greedy correctness; Qwen3 and the Llama-2 IOI comparison use vLLM greedy correctness. Eligibility stays fixed when a different backend or clean repeat gives a different answer. Knockout results therefore measure dependence conditional on the original baseline success. Generation-task eligibility requires actual generated answers or programs; teacher-forced reference scores do not establish correctness.

B.2 Head selection and common knockout

PP uses the norm-matched layer-mean reference in Equation (6) at strength one, with full downstream recomputation. AP substitutes corrupted activations at the answer-prediction position for controlled tasks and at continuation-prediction positions for generation-task localization, holding later attention outputs fixed. Rankings use signed mean damage, with layer and head indices breaking ties.

Both rankings are evaluated with the same mean-activation knockout on the same held-out cohort. Selected projection-input head slices are replaced jointly, at every prompt position and throughout decoding, with normal downstream recomputation. Controlled tasks use clean discovery-token means; direct generation tasks use means from correct discovery continuations. Transfer retains the source-task rankings and means, with separately stated comparisons using target-calibrated means.

GSM8K. Direct localization and arithmetic transfer screen the official test split under greedy eight-shot generation with a 512-new-token cap and freeze each model's completed-correct cohort. Generation stops at `Question:` or EOS. Direct localization uses 64 correct training questions, the model's own fixed correct continuations and two distinct aligned corruptions that change only digits in the final question. PP and the two AP conditions are ranked independently; the first corruption defines the primary AP comparison. Arithmetic transfer uses arithmetic-source rankings and means. Its separate target-calibrated comparison keeps those rankings and fits replacement means to correct GSM8K calibration generations.

HumanEval and MBPP transfer. Direct HumanEval localization screens disjoint pools of 64 discovery and 100 evaluation tasks, retaining correct programs in each. MBPP-to-HumanEval screens 374 MBPP training tasks and all 164 HumanEval tasks separately, then transfers rankings and source-completion means from eligible correct MBPP programs to the correct HumanEval cohort. MBPP questions become docstrings beneath reference-derived function signatures; tests are used only for grading. For code-task AP, a seeded mask replaces 15% of eligible docstring prose tokens, rounded up, while preserving token count, signatures and examples. Both prompts are scored against the same clean continuation. Llama, Qwen3.5 and Gemma4 MBPP source stubs end with a newline; Qwen3 retains its declared prompt format. Functional pass@1 uses the original tests, with identical programs for the same task required to receive consistent verdicts across conditions.

Generation evaluation produces new outputs, including $K = 0$ clean repeats; errors remain in the original frozen denominator. Target NLL and named logit margins are unavailable for these generation comparisons.

Numerical precision. Discovery and secondary full-logit scoring use Transformers; generation uses vLLM. GPT-2 and Llama-2 arithmetic discovery use FP32; Llama-2 IOI and expanded-model studies use BF16. Optimized-versus-sequential BF16 scores exceed the declared scalar tolerances in several studies, reaching differences of 0.625 IOI logit-margin units and 0.062271 arithmetic probability units in the Qwen3-4B/8B comparisons. Backend-dependent greedy prediction differences also occur. Bootstrap intervals describe sampling uncertainty and exclude these numerical execution differences.

B.3 Metrics, controls and uncertainty

For a controlled-task prompt of length T, scoring uses logits at position $T - 1$ for the complete target token at T. IOI's competitor is the same prompt's subject token in both states. Arithmetic discovery uses correct-token probability. Accuracy is full-vocabulary greedy target-token equality; target NLL is the negative log probability of that token. Positive NLL damage is an increase, while positive probability or margin damage is a decrease. Per-instance values are averaged across the frozen example population. Unavailable endpoints are left missing rather than assigned zero.

Candidate probability and conditional choice. For a task's candidate-token set $\mathcal{C}$, let p_v be the model's probability of token v and $y \in \mathcal{C}$ the correct answer. Define

$$m = \sum_{v \in \mathcal{C}} p_v, \qquad q = \frac{p_y}{m}, \qquad p_y = mq. \tag{14}$$

For IOI, $\mathcal{C}$ contains the recipient and subject names; for arithmetic, it contains the valid one-token answers 2–9. Thus m measures probability assigned to the candidate set and q measures preference for the correct answer within that set. Each is computed per prompt and then averaged. Mean q is not a ratio of aggregate probabilities, and multiplying mean m by mean q need not recover mean target probability. All cohort examples contribute, including those whose greedy output lies outside $\mathcal{C}$.

For IOI, let n be recipient-token NLL and $r = z_{\mathrm{IO}} - z_{\mathrm{S}}$ the recipient-minus-subject logit margin. When both scores use the same execution state and fixed labels,

$$p_{\mathrm{IO}} = e^{-n}, \qquad p_{\mathrm{S}} = e^{-n-r}, \qquad q = \mathrm{sigmoid}(r). \tag{15}$$

The sigmoid form computes conditional preference without dividing probabilities that may underflow. The corresponding arithmetic quantity is computed from logits normalized within its candidate set.

Generated-answer scoring. The GSM8K answer reader selects the last generated numeric candidate before the first stop delimiter or EOS, rather than requiring a generated #### marker. Reference answers are extracted after their final #### marker. Signed integers and decimals are canonicalized exactly, including valid comma grouping and redundant zeros. Unsupported fractions/scientific notation, malformed or ambiguous final numbers, empty outputs and stop boundaries that split a token are scored incorrect. Completed-correct screening additionally requires termination before the generation cap. Direct GSM8K knockout results report last-number accuracy: an unfinished output may still contain the correct final number. This is distinct from the stricter completed-correct diagnostic.

Head budgets and random controls. Controlled-task budgets include every integer from 0 through 16, then $20, 24, 28, 32, 40, 48, 56, 64$, with 96 and 128 added for arithmetic. Eight layer/count-matched random sets exclude the selected heads at $K = 8, 16, 32, 64$ and, for arithmetic, 128. Random draws may overlap each other. Infeasible controls remain unavailable. Arithmetic-to-GSM8K transfer uses every $K = 0, \ldots, 16$, with eight controls per method at $K = 8, 16$. The extended generation curves retain their study-specific budgets and controls alongside the primary $K = 0$–16 summary.

Curve summaries and uncertainty. AUC uses trapezoidal integration divided by the declared K interval length, including division by 16 for the primary summaries. Lower retained-accuracy AUC means greater disruption. With a shared baseline and integration range, the PP-minus-AP accuracy-damage AUC is the negative of the PP-minus-AP retained-accuracy AUC. Positive damage-AUC contrasts therefore favor greater PP disruption. NLL increase, probability damage and logit-margin damage retain their own units.

Controlled-task intervals use 10,000 paired bootstrap resamples of name or operand families, retaining example weights within each resampled family. Arithmetic-to-GSM8K intervals use 2,000 paired question resamples. Direct HumanEval and MBPP-to-HumanEval use 10,000 paired example resamples. Direct GSM8K resamples target questions jointly across conditions; the Qwen3-4B study uses 10,000 draws and the cross-model results retain their saved study-specific intervals. All are 95% percentile intervals. Control contrasts first average the realized random draws per example. Intervals are pointwise, conditional on frozen rankings and controls; they do not measure uncertainty from repeating discovery. Random-set variation and sampling uncertainty are reported separately.

C Controlled-Task Analyses

C.1 Motivating IOI example and confirmation

Intervention and metric. Figure 1 uses float32 GPT-2 Small. At the final prompt position t, we replace a selected head's bias-free post-projection write $\mathbf{a}_{\ell h,t}$. The replacement comes either from the name-swapped input or from a clean-input execution with that head's output weights zeroed:

$$\mathbf{a}_{\ell h,t} \leftarrow \begin{cases} \mathbf{a}_{\ell h,t}(x_{\mathrm{cf}}) & \text{activation substitution,} \\ \mathbf{a}'_{\ell h,t}(x) & \text{parameter-side substitution.} \end{cases} \tag{16}$$

The supplementary query/key-zeroing condition uses the latter construction with the query and key weights zeroed instead. QKV biases and the shared output bias remain unchanged. Later attention outputs are fixed at clean values; MLPs and normalization recompute. Thus this illustration compares local substituted writes, rather than full-execution PP.

The clean prompt is *When Mary and John went to the store, John gave a bottle of milk to*, with intended completion Mary; the corrupted prompt swaps Mary and John. With fixed clean labels, r is the Mary-minus-John logit margin, and figure damage is

$$d_{\mathrm{norm}}(\%) = 100\,\frac{r_{\mathrm{clean}} - r_{\mathrm{patched}}}{r_{\mathrm{clean}} - r_{\mathrm{cf}}}. \tag{17}$$

It is relative to the clean-minus-corrupted margin gap, not accuracy loss; negative values and values above 100% are possible. Activation substitution causes 42.8% damage for L9.H9 versus 15.4% for L9.H6; output-weight zeroing reverses this ordering to 4.8% versus 15.9%. Zeroing the output weights gives a zero write and therefore no direction-based vocabulary rank.

Independent confirmation. L9.H6, L9.H9 and L10.H0 were fixed before development. Seven templates and disjoint name-pair families yield 160 development and 224 confirmation candidates, of which 142 and 204 are initially correct. Confirmation contains 54 name-pair families; templates 5–6 are held out of development. The development sweep of all 144 heads finds no additional head with mean absolute input-substitution effect at most 0.1 and mean parameter-side damage at least 0.2 in normalized units. Confirmation is not searched for new cases.

Confirmation reports signed normalized margin changes $-d_{\mathrm{norm}}/100$, averaging per-prompt ratios. Every denominator exceeds 10^{-6}. For H6/H9, mean changes are $-0.295/-0.753$ under input

substitution, $-0.140/-0.366$ under output zeroing and $-0.158/-0.422$ under query/key zeroing: H9 has the larger mean effect in every condition. Subtracting the H9-minus-H6 change under output zeroing from that under input substitution gives -0.232 (95% interval $[-0.253, -0.211]$). It is -0.219 $[-0.243, -0.196]$ on 149 seen-template prompts and -0.267 $[-0.309, -0.226]$ on 55 held-out prompts. Intervals use 10,000 paired name-family resamples and are pointwise. The example's ordering reversal therefore illustrates prompt-specific behavior; confirmation supports a difference in relative effect magnitude.

J-Lens readouts (Gurnee et al., 2026) are observational. They use after-block Jacobian–vector products and a float32 GPT-2 bank fitted on 64 WikiText windows with target layer 10, without fitting on IOI. Calibration-domain differences and the local approximation limit semantic interpretation. Both confirmation heads track the current recipient under name swaps; their behavior does not establish distinct task-wide head types.

C.2 Common-knockout comparisons

Table 6 and Figure 6 give the six controlled-task comparisons underlying the matched-control claim in Section 4.2. Within each model/task, both selections use the same frozen evaluation cohort and all-position activation-mean knockout. Cohorts and precision are specified in Table 5; values are compared within cells, not as precision-matched cross-model estimates.

Model	Setting	N	AP	PP	PP − AP (95% CI)
GPT-2 Small	IOI	454	**0.4084**	0.5747	$+0.1663[+0.1412, +0.1918]$
Llama-2-7B	Arithmetic	77	0.3937	**0.0418**	$-0.3519[-0.4559, -0.2722]$
Qwen3-4B	IOI	346	**0.5801**	0.6571	$+0.0770[+0.0464, +0.1078]$
Qwen3-4B	Arithmetic	66	0.5677	**0.3830**	$-0.1847[-0.2832, -0.0804]$
Qwen3-8B	IOI	364	**0.5914**	0.7400	$+0.1486[+0.1224, +0.1736]$
Qwen3-8B	Arithmetic	75	0.7188	**0.5938**	$-0.1250[-0.1762, -0.0557]$

Table 6: Normalized retained-accuracy AUC over $K = 0$–16; lower values mean greater disruption. Bold shading marks lower AUCs. Positive PP-minus-AP differences indicate greater AP disruption. Intervals use 10,000 paired family-bootstrap draws.

Both selectors cause more accuracy damage than their own layer-matched random controls at $K = 8, 16$ in all six comparisons: all 24 paired 95% excess-damage intervals exclude zero. Each control estimate averages eight matched draws. The full curves retain nonmonotonic effects and budget-specific reversals. For example, GPT-2 IOI favors AP over the AUC window but PP at $K = 2$, where PP-minus-AP retained accuracy is -9.47 $[-12.42, -6.64]$ percentage points; at $K = 3$, the contrast is unresolved, -1.10 $[-4.66, 2.46]$. These are pointwise intervals, conditional on the frozen rankings. All six AUC orderings persist over their full declared ranges.

The Llama-2-7B IOI result in Table 1 uses BF16 and the same ABC corruption. Only 16 of 1,024 candidates satisfy tokenization and alignment requirements, leaving five initially correct discovery and five evaluation examples. On this small cohort, retained-accuracy AUC is 0.056 for PP versus 0.631 for AP, with paired difference -0.575 $[-0.875, -0.275]$. GPT-2 Small answers none of the 180 arithmetic candidates correctly, so its arithmetic AUC is unavailable, not zero. These eligibility limits constrain the scope of the model comparisons.

Accuracy loss alone does not identify a task-specific computation: Llama-2 arithmetic PP knockout at $K = 1$ yields zero accuracy and 59/77 nonnumeric outputs. Numerical discrepancies and their implications are reported in Appendix B.2.

C.3 Candidate probabilities and head-group effects

Candidate mass m and conditional correct-choice probability q are defined in Appendix B.3. We compare IOI's two candidate names and arithmetic's valid one-token answers (2–9), using the same BF16 full-logit state for both quantities. Figure 3 contrasts full top-eight selections with their shared and selector-exclusive subsets.

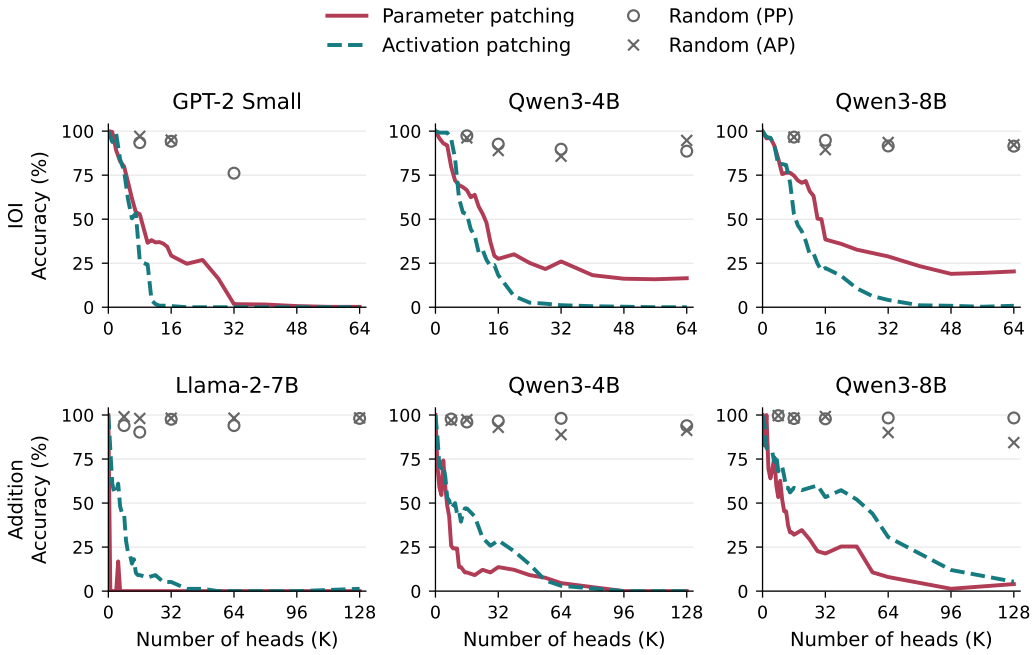


Figure 6: Full ranked-accuracy curves for the six controlled-task comparisons. Gray markers average layer-matched random controls. IOI extends to 64 heads and arithmetic to 128.

Fixed-budget comparisons. The figure and Table 7 retain examples correct under both the original native baseline and the intact full-logit scorer: 339/346 and 343/346 IOI examples for Qwen3-4B and Qwen3.5-4B, plus all 66 and 81 arithmetic examples. No intervention outcome changes these subsets. Exclusive groups contain five/six heads in Qwen3/Qwen3.5 IOI and seven in either arithmetic model; shared groups contain three/two and one/one heads. Thus exclusive PP/AP groups have matched counts but are smaller than the full eight-head sets.

Boxes show medians and quartiles; whiskers span the 5th–95th percentiles. The wrong-name-share and valid-answer-probability axes omit some outlying dots from view, but all observations enter the summaries and all boxes and whiskers remain visible. Paired contrasts compare intervened candidate mass m and incorrect-candidate share $1 - q$. Pointwise 95% intervals use 10,000 family resamples, conditional on the frozen rankings.

Model/task	Contrast	Candidate probability	Wrong share among candidates
Qwen3 IOI	Full PP–AP	$+.103$ $[.087, .119]$	$+.137$ $[.117, .156]$
	Exclusive PP–AP	$+.092$ $[.074, .110]$	$+.016$ $[.010, .022]$
Qwen3.5 IOI	Full PP–AP	$+.354$ $[.338, .370]$	$+.127$ $[.115, .138]$
	Exclusive PP–AP	$+.395$ $[.378, .411]$	$+.063$ $[.055, .070]$
Qwen3 arithmetic	Full PP–AP	$-.001$ $[-.041, .030]$	$+.095$ $[.036, .148]$
	Exclusive PP–AP	$-.026$ $[-.074, .009]$	$+.200$ $[.125, .257]$
Qwen3.5 arithmetic	Full PP–AP	$-.028$ $[-.053, -.007]$	$+.047$ $[.015, .080]$
	Exclusive PP–AP	$-.029$ $[-.052, -.012]$	$+.051$ $[.017, .083]$

Table 7: PP-minus-AP probability differences for full and exclusive head sets at $K = 8$, with paired 95% family-bootstrap intervals. Positive values indicate more candidate probability or a larger incorrect-candidate share under PP.

Shared-head knockouts also increase wrong-candidate share in all four cells (by .058/.014 for Qwen3/Qwen3.5 IOI and .078/.027 for arithmetic; each paired interval excludes zero). Full-set effects need not equal the sum of shared and exclusive effects. A separate native check on all 346 IOI

examples per model confirms greater choice damage from PP-exclusive than AP-exclusive groups: .025 versus .010 in Qwen3 and .067 versus .005 in Qwen3.5. The PP groups also exceed their own layer/unit-kind-matched controls. AP-exclusive groups remove more candidate-name probability, while every new group-knockout error falls outside the two-name pair. The observed preference shift therefore does not identify a subject-copy mechanism in these groups.

Full-curve comparison. Table 8 uses all originally native-correct examples, including the seven/three IOI examples with different full-logit baseline predictions. Full-logit scores evaluate $K = 0$–16; native generation continues to define accuracy. Here the contrast is PP-minus-AP *damage*, so positive mass or choice contrasts have the opposite sign to differences in intervened probabilities.

Model	Task	Target NLL	Candidate mass	Choice
Qwen3-4B	IOI	$-.287$ $[-.348, -.230]$	$-.083$ $[-.092, -.074]$	$+.136$ $[.123, .150]$
Qwen3-4B	Arithmetic	$+.421$ $[.282, .556]$	$+.006$ $[-.011, .027]$	$+.081$ $[.026, .126]$
Qwen3.5-4B	IOI	-1.768 $[-1.840, -1.695]$	$-.234$ $[-.243, -.225]$	$+.033$ $[.021, .045]$
Qwen3.5-4B	Arithmetic	$+.011$ $[-.060, .084]$	$+.001$ $[-.008, .014]$	$-.005$ $[-.039, .028]$

Table 8: PP-minus-AP damage AUC over $K = 0$–16 under common full-logit knockouts. Positive values indicate greater PP damage. Intervals are conditional on the frozen discovery rankings.

The IOI AUCs resolve greater PP conditional-choice damage and greater AP candidate-mass damage in both models. This ordering need not hold at every budget: Qwen3.5 at $K = 16$ has nearly equal choice means (.689 PP, .683 AP). The native Qwen3 IOI analysis also finds a mass-ordering reversal: PP retains less name probability at $K = 1$–5 and more at $K = 6$–16. For arithmetic, both mass intervals include zero; Qwen3.5's NLL and choice intervals also include zero despite its resolved fixed-$K = 8$ comparison. These results describe answer distributions, not a shared arithmetic algorithm.

C.4 Individual-head readouts and knockout effects

Protocol and case selection. The Qwen3-4B cases in Figures 4 and 7 come from the frozen union of PP/AP top-eight selections. Individual BF16 knockouts replace a head's write by its discovery mean at all prompt positions, with normal downstream recomputation, on 346 initially correct IOI and 66 arithmetic examples. Readouts separately use the first 24 shared baseline-correct prompts per task and both FP32 J-Lens and plain lenses. At position t, a readout scores a possible continuation at $t + 1$; only the final site scores the task answer. These observational scores are neither intact-model probabilities nor causal effects of that position. The figures use a common zero-centered scale; dots denote leading spaces and superscripts distinguish repeated name occurrences.

IOI. The displayed PP-only L26.H4, AP-only L32.H8 and shared L30.H7 rank Steven 133rd, first and fourth at the final site. The recipient reaches their final top four on 1/24, 22/24 and 7/24 readout prompts. Their individual knockouts cause 12/346 (3.5%), 3/346 (0.9%) and 33/346 (9.5%) new errors; all three preserve the displayed prompt's answer. L26.H4 was selected after inspecting the top-eight PP-only heads for a clearer name-category readout. Another PP-only head, L22.H30, has no final recipient top-ten readout on the panel but causes 36/346 errors. Thus answer visibility does not order these heads' knockout effects.

Arithmetic. PP-only L35.H27, AP-only L34.H28 and shared L35.H26 cause 19/66 (28.8%), 5/66 (7.6%) and 8/66 (12.1%) new errors. Every L35.H27 error copies an operand; its paired 95% family-bootstrap accuracy damage is .288 [.140,.405], while its same-layer control L35.H30 causes no errors. The correct sum reaches the final top four on 0/24, 21/24 and 24/24 readout prompts, respectively; it reaches L35.H27's bottom four on 10/24. The head therefore affects producing the sum without a prominent answer readout, but this does not identify the computation it performs.

Scope of the readouts. These selected cases do not define PP/AP head types. A broader prespecified 57-head comparison finds no stable representational signature. In Qwen3.5, the first two PP arithmetic

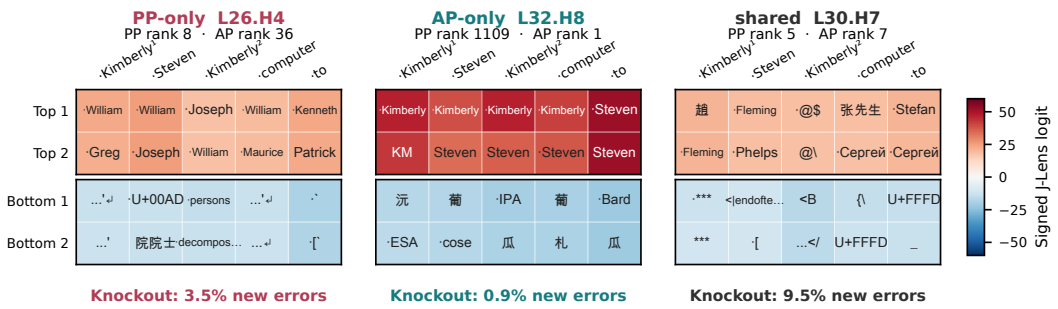


Figure 7: IOI head readouts and knockout effects in Qwen3-4B, with Steven as recipient and Kimberly as repeated subject. Readout encodings follow Figure 4; knockout rates use the initially correct IOI evaluation set. All three knockouts preserve the displayed prompt's answer.

units individually cause no errors, another PP-exclusive head is answer-readable, and the top AP head is not. The answer-readable AP head L31.H8 causes no accuracy errors despite an NLL increase of .032 [.028,.039]. These counterexamples distinguish readability, probability change and accuracy dependence.

J-Lens calibration includes unresolved finite-difference discrepancies: four Qwen3 checks have 0.40–0.75% relative L_2 error and one Qwen3.5 check has 17.0%. They test single-article L0-to-final-layer Jacobians, not the averaged banks used here. Fitted-transport readouts remain exploratory; late-layer arithmetic cases use identity transport. Independent knockout measurements do not validate lens transport.

D Complex Capabilities and Transfer

The direct and transfer comparisons evaluate independently derived PP and AP rankings with the same held-out knockout. Evaluation uses model-specific, frozen baseline-correct cohorts; clean-repeat errors remain in the denominator. Accuracy damage is clean-repeat accuracy minus intervention accuracy. The reported contrast is PP-minus-AP damage AUC over $K = 0$–16, so positive values indicate greater accuracy loss from knocking out PP-selected units. Intervals are paired within each model and task. These comparisons measure dependence on selected units under knockout.

D.1 Direct GSM8K and HumanEval localization

GSM8K evaluates last-number accuracy from greedy generations; HumanEval evaluates functional pass@1 under the original tests. The GSM8K comparison favors PP in all but Qwen3.5-4B. Nine HumanEval comparisons resolve toward PP, while the remaining five are unresolved (Table 9). The small HumanEval cohorts, including two with only two programs, limit comparisons across checkpoints. Target NLL and named logit margins are unavailable for these generation studies.

D.2 Cross-task transfer

Arithmetic-to-GSM8K uses arithmetic rankings and arithmetic-discovery means. MBPP-to-HumanEval uses code-task rankings and source-completion means. Neither study reselects units on the target evaluation cohort. Paired intervals favor greater PP disruption in 10 of 14 arithmetic-to-GSM8K comparisons and all 14 MBPP-to-HumanEval comparisons (Table 10). Arithmetic transfer favors AP for Qwen3.5-4B and Gemma4-E4B, while Qwen3-4B and Qwen3.5-2B remain unresolved. These results establish the influence of source-selected units on related tasks.

D.3 Generation trajectories and failure forms

Generation accuracy across knockout sizes. The Qwen3-4B generation curves use official HumanEval source stubs and an eight-shot GSM8K prompt. Both retain all initially correct evaluation examples, including clean-repeat errors at $K = 0$. The Qwen3-4B HumanEval $K = 0$–16 PP–AP

Task	Model	N	ΔAUC	Paired 95% interval
GSM8K	Llama-2-7B	212	**+.1790**	[+.1337, +.2242]
	Llama-3-8B	724	**+.2305**	[+.2070, +.2528]
	Llama-3.1-8B	747	**+.2040**	[+.1836, +.2265]
	Llama-3.2-1B	116	**+.1237**	[+.0832, +.1673]
	Llama-3.2-3B	449	**+.3802**	[+.3517, +.4085]
	Qwen3-0.6B	799	**+.7113**	[+.6898, +.7326]
	Qwen3-1.7B	967	**+.8185**	[+.8023, +.8345]
	Qwen3-4B	1,163	**+.1777**	[+.1578, +.1982]
	Qwen3-8B	1,186	**+.0832**	[+.0665, +.1012]
	Qwen3.5-0.8B	630	**+.2012**	[+.1901, +.2127]
	Qwen3.5-2B	887	**+.0775**	[+.0672, +.0873]
	Qwen3.5-4B	1,115	**−.0613**	[−.0751, −.0482]
	Gemma4-E2B	327	**+.0934**	[+.0750, +.1117]
	Gemma4-E4B	786	**+.0529**	[+.0353, +.0703]
HumanEval	Llama-2-7B	2	**+.3594**	[.0000, +.7188]
	Llama-3-8B	21	**+.4717**	[+.3155, +.6205]
	Llama-3.1-8B	20	**+.1156**	[−.0704, +.3063]
	Llama-3.2-1B	2	**+.2188**	[.0000, +.4375]
	Llama-3.2-3B	8	**+.3516**	[+.2070, +.5039]
	Qwen3-0.6B	10	**+.2063**	[+.0844, +.3563]
	Qwen3-1.7B	31	**+.7470**	[+.6351, +.8478]
	Qwen3-4B	39	**+.1130**	[−.0136, +.2412]
	Qwen3-8B	57	**+.2615**	[+.1667, +.3558]
	Qwen3.5-0.8B	5	**+.4563**	[+.1063, +.8063]
	Qwen3.5-2B	15	**+.3667**	[+.2479, +.4875]
	Qwen3.5-4B	36	**+.1484**	[+.0200, +.2752]
	Gemma4-E2B	6	**−.0052**	[−.0833, +.0469]
	Gemma4-E4B	10	**+.2688**	[+.1374, +.4125]

Table 9: Direct localization on GSM8K and HumanEval. Positive PP-minus-AP damage AUC indicates greater PP disruption; shading identifies the more disruptive method. N counts initially correct evaluation examples.

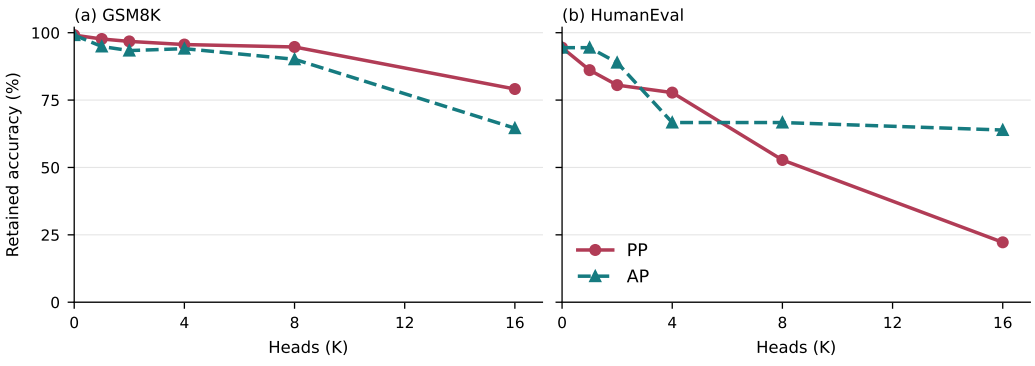


Figure 8: Qwen3.5-4B accuracy after direct localization. **Left:** GSM8K; **Right:** HumanEval. PP and AP selections use the common knockout.

Transfer	Model	N	ΔAUC	Paired 95% interval
Arithmetic → GSM8K	Llama-2-7B	208	**+.3433**	[+.2997, +.3870]
	Llama-3-8B	718	**+.5381**	[+.5137, +.5620]
	Llama-3.1-8B	736	**+.5493**	[+.5251, +.5729]
	Llama-3.2-1B	124	**+.2613**	[+.2243, +.3004]
	Llama-3.2-3B	447	**+.2046**	[+.1770, +.2317]
	Qwen3-0.6B	802	**+.3026**	[+.2862, +.3201]
	Qwen3-1.7B	619	**+.3462**	[+.3253, +.3673]
	Qwen3-4B	1,137	**+.0198**	[−.0016, +.0413]
	Qwen3-8B	1,158	**+.0270**	[+.0096, +.0438]
	Qwen3.5-0.8B	623	**+.0332**	[+.0284, +.0381]
	Qwen3.5-2B	891	**+.0047**	[−.0042, +.0134]
	Qwen3.5-4B	1,116	**−.0781**	[−.0897, −.0660]
	Gemma4-E2B	324	**+.1645**	[+.1396, +.1892]
	Gemma4-E4B	792	**−.1018**	[−.1122, −.0915]
MBPP → HumanEval	Llama-2-7B	22	**+.2500**	[+.1420, +.3636]
	Llama-3-8B	65	**+.2486**	[+.1587, +.3389]
	Llama-3.1-8B	58	**+.1875**	[+.0986, +.2770]
	Llama-3.2-1B	33	**+.2263**	[+.1297, +.3248]
	Llama-3.2-3B	44	**+.2621**	[+.1641, +.3644]
	Qwen3-0.6B	46	**+.5516**	[+.4429, +.6549]
	Qwen3-1.7B	82	**+.7370**	[+.6673, +.8026]
	Qwen3-4B	89	**+.2195**	[+.1643, +.2767]
	Qwen3-8B	109	**+.1046**	[+.0470, +.1626]
	Qwen3.5-0.8B	36	**+.4635**	[+.3776, +.5547]
	Qwen3.5-2B	50	**+.5700**	[+.4825, +.6550]
	Qwen3.5-4B	77	**+.1993**	[+.1343, +.2662]
	Gemma4-E2B	43	**+.3779**	[+.2762, +.4767]
	Gemma4-E4B	49	**+.3865**	[+.2908, +.4802]

Table 10: Cross-task knockout transfer using source-task rankings and replacement means. Positive PP-minus-AP damage AUC indicates greater PP disruption; shading identifies the more disruptive method. N counts initially correct target-task examples.

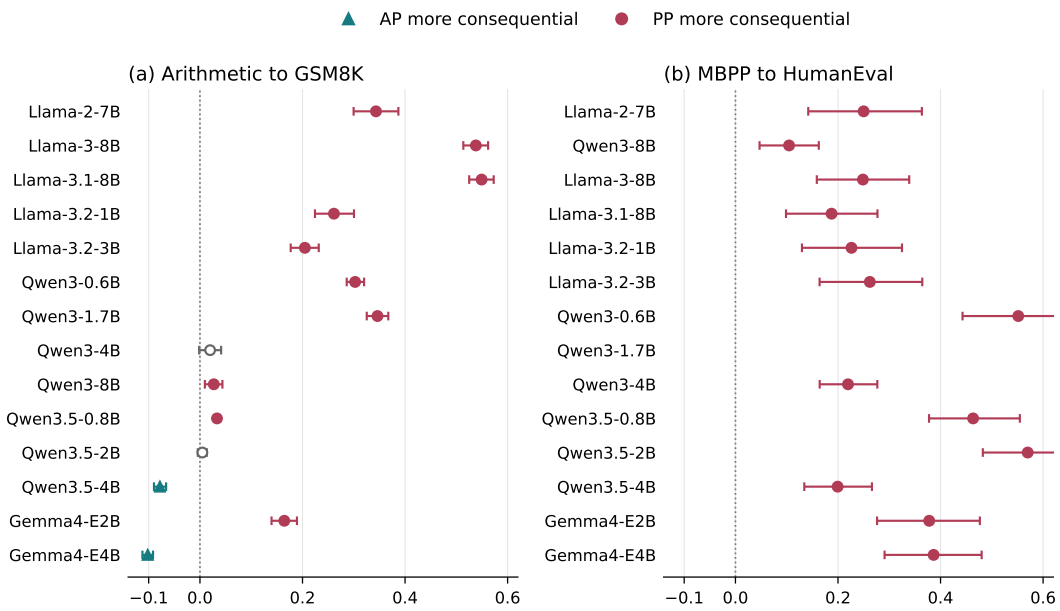


Figure 9: Transfer from arithmetic to GSM8K (a) and MBPP to HumanEval (b). Points show PP-minus-AP damage AUC over $K = 0$–16, with paired 95% intervals. Positive values indicate greater PP disruption; hollow markers denote intervals containing zero.

AUC interval includes zero, while its extended $K = 0$–64 interval favors PP. On GSM8K, PP has greater damage at $K = 2$–8 and 64, AP has greater damage at $K = 16$ and 32, and the $K = 0$–64 AUC interval includes zero (Figure 10). HumanEval accuracy is equal for PP and AP at $K = 1$; at $K = 64$, PP retains 0/39 correct programs and AP retains 31/39.

Code failures under transfer. At $K = 16$ in MBPP-to-HumanEval transfer, the visible program form differs across checkpoints. We parse the official HumanEval prompt plus the contextually decoded visible completion as Python code. A parse failure is a syntactic observation; functional failure is graded by the original tests. Counts use every example in each frozen target cohort (Table 11).

Error forms across knockout sizes. For each task, a fixed set of 12 originally correct Qwen3-4B examples is followed across every evaluated K (Figure 5). Arithmetic and IOI categories are deterministic. GSM8K and HumanEval labels combine language-model judgments with direct reading of outputs for which annotation failed. This mixed procedure supports illustrative descriptions of the fixed panel, not population semantic-error rates or a uniform semantic comparison between PP and AP. Its $K = 0$ shows the original correct screening outputs.

The panel illustrates task-dependent failure forms. Arithmetic shifts from correct sums toward operands earlier under PP; at $K = 128$, both selectors have no correct sums and identical category counts. In IOI at $K = 16$, PP produces the repeated subject on 3 prompts, while AP produces function words or pronouns on 11, consistent with the separate probability analysis (Section 4.2). In HumanEval, late PP errors shift toward invalid or incomplete programs; AP retains 10 functional programs at $K = 64$. GSM8K errors usually retain numerical answers, with a less uniform path across K. Both selectors also show error-to-correct transitions in every task, including 10 PP and four AP events in arithmetic. These cumulative knockouts do not isolate the newly added head's effect. Random generations were not classified at every K; their GSM8K and HumanEval accuracy controls remain in Figure 10.

Nonmonotonic knockout effects. For PP-selected Qwen3-4B heads on GSM8K, accuracy rises from 40.1% at $K = 8$ to 57.4% at $K = 16$, then falls to 44.2% at $K = 32$. These comparisons

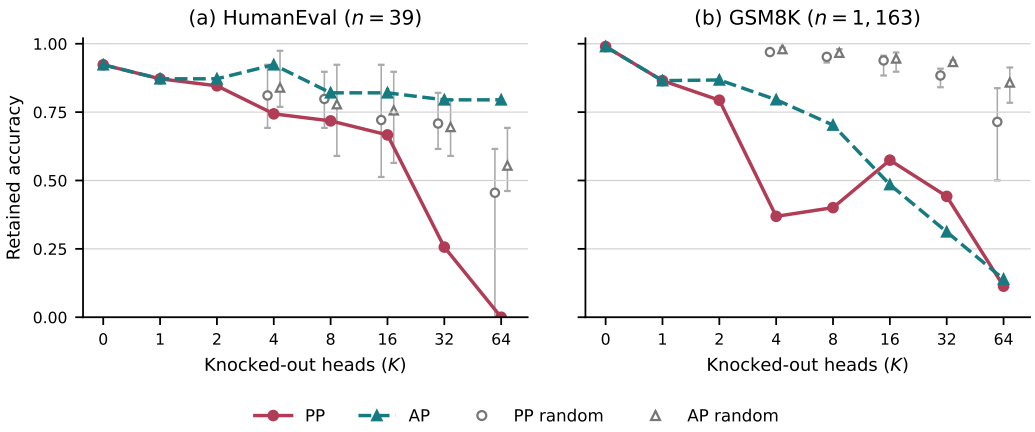


Figure 10: Qwen3-4B generation accuracy after head knockouts on HumanEval (a) and GSM8K (b). Gray markers show means of eight random controls per selector and whiskers their range. $K = 0$ is a clean repeat; head counts are displayed at equal spacing despite unequal increments. Paired AUC intervals appear in Table 9 for $K = 0$–16.

		Functional failures		Unparseable		Random range (unparseable)	
Model	n	**PP**	**AP**	**PP**	**AP**	**PP**	**AP**
Qwen3-4B	89	65	11	52	6	0–4	0–4
Qwen3-8B	109	33	22	2	4	1–8	0–3
Qwen3.5-4B	77	41	19	20	0	0–1	0–2

Table 11: Code failures after MBPP-to-HumanEval transfer at $K = 16$. Unparseable outputs fail Python AST parsing. Random ranges span eight layer-matched control sets per method.

use the same evaluation examples and nested head sets; strict completed-answer scoring follows the same pattern. Further removals can change the balance of surviving contributions and the subsequent generation trajectory, offering a possible explanation for this response. Chen et al. (2026) identify distinct circuits that can each support the same task. These findings provide relevant context; the mechanism behind the present rebound remains unresolved.

E Steering and Pinpoint Training

Steering and pinpoint training test whether localized heads provide useful targets for improving GSM8K accuracy. Results in Table 3 use zero-shot greedy generation and last-number scoring on all 1,319 official test questions, with a 512-new-token cap and stopping at `Question:` or EOS. All questions remain in the denominator, including capped outputs and parsing failures. Its single Base column displays the lower recorded unmodified repeat; paired inference uses each study's original controls. The SWE-rebench experiment is described separately in Section 4.4.

Steering strengths and training learning rates were selected using PP and shared across selectors.

E.1 Inference-time steering

The GSM8K comparison uses independently frozen task-native PP/AP selections of $K = 16$ units. As in contrastive activation addition (Rimsky et al., 2024), we average activation differences between paired prompt conditions. For the direction-fitting set $\mathcal{D}_{\text{fit}}$, the direction at layer ℓ is

$$\mathbf{v}_\ell = \frac{1}{|\mathcal{D}_{\text{fit}}|} \sum_{x \in \mathcal{D}_{\text{fit}}} \left[\mathbf{a}_\ell^{(8)}(x) - \mathbf{a}_\ell^{(0)}(x) \right], \tag{18}$$

where $\mathbf{a}_\ell^{(8)}(x)$ and $\mathbf{a}_\ell^{(0)}(x)$ are the final-prompt-token activations under eight-shot and zero-shot prompting. During generation, the selected slices receive the scaled direction at every prompt and decode token. The layer-matched random control matches PP's layer histogram. Steering calibration uses disjoint questions.

Paired comparisons favor PP over its original baseline and random for Qwen3-4B and Llama-3-8B. Their gains over random are 11.60 accuracy points (95% interval $[8.81, 14.39]$) and 6.52 ($[3.82, 9.22]$), respectively. Qwen3-1.7B and Llama-2-7B exceed random by 2.96 ($[0.58, 5.33]$) and 1.21 ($[0.06, 2.36]$) points, while their baseline comparisons remain unresolved. Qwen3-4B has a resolved PP-minus-AP advantage: 7.88 points ($[5.23, 10.53]$).

E.2 Pinpoint training

Training protocol. The fourteen-model study uses 6,961 GSM8K supplied-target training questions and 512 held-out training questions, split equally between learning-rate selection and independent comparison. Each arm starts from the same Base checkpoint; PP and AP use their own frozen GSM8K localization rankings. Training uses bfloat16, global batch 32 and at most one epoch (218 updates). The learning-rate grid is $\{10^{-5}, 5 \times 10^{-5}, 10^{-4}\}$, evaluated on the first 256 questions with top-16 output-weight training. Both AP and random controls train $K = 16$ output-weight slices for 218 updates. The random mask matches AP's layer, physical unit kind and trainable coordinate counts, with one draw per model from all eligible units in the corresponding layers and unit kinds, including ranked heads.

For a selection $\mathcal{S}$, a binary mask $\mathbf{m}_\mathcal{S}$ over distinct scalar parameter coordinates defines the trained checkpoint:

$$\boldsymbol{\theta}_{\text{tuned}} = \boldsymbol{\theta}_0 + \mathbf{m}_\mathcal{S} \odot \delta\boldsymbol{\theta}, \tag{19}$$

where $\boldsymbol{\theta}_0$ is the original checkpoint, $\delta\boldsymbol{\theta}$ is the optimizer's update and $\odot$ denotes elementwise multiplication. Parameters outside the mask remain frozen. Equal head counts can select different numbers of scalar coordinates.

Full fine-tuning and resource costs. Full SFT uses the same training questions, batch size and 218 updates, with a fixed learning rate of 5×10^{-6}. It updates all non-embedding text parameters, freezing input embeddings, including Gemma's per-layer embeddings. Optimizer state remains on the training devices; multiple-device runs use ZeRO-1. Table 12 reports its accuracy and compares resource costs with the common $K = 16$, 218-update PP recipe. PP uses less peak device memory in all fourteen models and fewer device minutes in ten; Qwen3-1.7B and the three Qwen3.5 models take more device minutes. These measurements cover training, excluding loading, checkpoint export, evaluation and recipe search.

F Discussion

PP and AP expose dependence under different interventions, including different propagation and positional support. Common knockouts compare the resulting head selections, while constructive studies test their usefulness for adaptation. Whether these findings extend to other behaviors, replacement references and update budgets remains open.

		Full SFT	Device min		Peak GiB		Full SFT
Family	**Size**	**Acc. (%)**	**PP**	**Full SFT**	**PP**	**Full SFT**	**Devices**
Llama	2-7B	34.8	6.01	48.06	25.74	65.59	8
	3-8B	61.9	5.93	51.74	18.17	76.97	8
	3.1-8B	62.0	5.98	51.39	30.54	76.97	8
	3.2-1B	21.1	2.78	3.07	3.89	24.35	1
	3.2-3B	46.3	5.01	11.48	12.51	56.32	1
Qwen3	0.6B	52.8	6.63	21.21	3.25	10.12	1
	1.7B	68.2	6.95	6.15	5.64	33.74	1
	4B	82.0	8.28	14.27	15.51	45.64	4
	8B	81.9	7.58	60.72	18.43	78.54	8
Qwen3.5	0.8B	51.2	18.82	6.65	4.74	19.89	1
	2B	63.4	11.75	7.04	7.66	37.00	1
	4B	83.1	18.99	15.85	12.76	48.29	4
Gemma4	E2B	34.6	13.21	39.36	15.00	31.38	4
	E4B	53.4	14.72	79.16	30.30	32.18	8

Table 12: Full-SFT GSM8K accuracy and training costs at a common 218-update budget. Costs compare top-16 output-weight PP with full SFT; they exclude recipe search. Device minutes sum training time across PPUs; peak memory is per device. PP uses one PPU.