WHAT CARRIES MATHEMATICAL COMPUTATION? TRACING FUNCTIONAL CHANGES IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern language models exhibit strong mathematical capabilities, yet fine-tuning on specific tasks can compromise performance on other mathematical problems. Isolating newly acquired mathematical capabilities into compact updates is therefore important for preserving them while limiting interference with existing capabilities. We propose SAE-guided update selection, which connects changes in internal representations to the selection of parameter updates. A shared sparse autoencoder (SAE) identifies feature changes between the base and fine-tuned models, and these changes are used together with prediction loss to score rank-one terms of the LoRA update under a fixed storage budget. Across four model pairs from the Qwen, Gemma, and Llama families, feature-based scoring consistently improves over prediction-loss scoring. With only 25% of the full adapter's factor storage, the joint criterion surpasses Full LoRA in target-task accuracy on three pairs by 3.2–6.3 percentage points, while also preserving more GSM8K accuracy on the same pairs. Beyond selection performance, we test whether the identified features contribute to computation itself. Removing them impairs later computation even after supplying the correct next token. Moreover, restoring carry-related features to their base values reduces final-answer accuracy by 26.9 points on carry-dependent problems. Together, these results connect compact parameter updates, representation changes, and internal features that support successive mathematical computation.

1 INTRODUCTION

Large language models have made substantial progress on mathematical reasoning. On FrontierMath Tier 4 (v2), GPT-6 Astra achieves a reported score of 97.6% (OpenAI, 2026), while Epoch AI reports 87.8% for Claude Fable 5.1 (Epoch AI, 2026). These results demonstrate strong performance on challenging mathematical problems, but benchmark scores alone do not reveal which internal changes support a particular mathematical capability. Fine-tuning can induce many changes in model parameters and representations, only some of which may be necessary for the acquired capability. Identifying these changes could enable the capability to be retained in a compact parameter update, reduce interference with performance beyond the training tasks, and reveal how the corresponding internal representations participate in the computations that produce a solution. This raises a question: **Can we isolate a compact set of parameter changes that preserves a learned mathematical capability, and identify how the associated representations support computation?**

We study this question by comparing language models before and after fine-tuning on mathematical tasks, where the learned parameter update is directly available for analysis. This setting allows us to examine both which parts of the update are sufficient to preserve task performance and how the associated representation changes contribute to the computation itself.

Mathematical reasoning is particularly suitable for studying this problem because solutions often proceed through a sequence of dependent computations, where the result of one step directly affects those that follow. In column addition, for example, a carry produced in one column becomes an input to the next. An internal representation may therefore contribute not only to the current prediction, but also to later computation. As a result, correct next-token prediction alone does not establish that the

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.

model can complete the remaining solution. We consequently ask two complementary questions: **1) Which parameter changes preserve the acquired mathematical capability? 2) Which changed features affect subsequent computation even when the immediate prediction is correct?**

Prior work provides complementary tools for studying these questions. Mechanistic studies have identified components involved in numerical comparison and arithmetic (Hanna et al., 2023; Nikankin et al., 2025). Activation patching tests the causal effects of internal representations on model outputs (Zhang & Nanda, 2024), while task arithmetic manipulates parameter differences to transfer learned behavior (Ilharco et al., 2022). However, these approaches primarily examine representation-level effects or parameter-level changes separately. Our setting requires connecting the two: identifying which fine-tuning-induced parameter changes preserve mathematical performance and whether the associated representation changes support subsequent computation.

To identify which fine-tuning changes preserve the acquired capability, we use a shared sparse autoencoder (SAE) to represent the base and fine-tuned models in common feature coordinates. Fine-tuning-induced feature changes, together with prediction loss, are then used to score and select singular components of the LoRA update (Hu et al., 2021) under a fixed storage budget. We study column addition, column subtraction, and variable isolation, whose verified intermediate states also support later intervention analysis. Across four model pairs, feature-based scoring consistently outperforms prediction-loss scoring. With only 25% of the full adapter's factor storage, the combined criterion surpasses Full LoRA on target-task accuracy for three pairs while retaining high GSM8K accuracy. These results show that **feature changes can identify a compact subset of the update that preserves acquired mathematical performance**.

We next ask whether the identified representations contribute to computation beyond producing the immediate prediction. We delete selected features and externally supply the correct next token before generation continues. Despite this correction, subsequent states and final answers deteriorate, showing that the effects of these features persist beyond the current prediction. Thus, **the identified features support subsequent computation rather than merely improving next-token prediction**.

Finally, we examine a specific computational role by restoring carry-related features in the selected model to their Base values while leaving the parameter update unchanged. This intervention selectively disrupts carry-dependent solutions, providing more direct evidence that the retained feature changes participate in carry computation. Together, these analyses connect effective parameter selection with the internal representations that support successive computation.

Our contributions are:

- We introduce **SAE-guided update selection**, which uses shared SAE features and path-averaged scoring to prioritize rank-one LoRA directions based on both prediction and feature-recovery effects, enabling compact update selection under a fixed budget.
- We show that **acquired mathematical capabilities can be retained by a compact subset of the fine-tuning update**, with feature changes providing a stronger selection signal than prediction loss alone. At 25% of the adapter's factor storage, the joint criterion outperforms the full adapter on both target tasks and GSM8K for three of four model pairs.
- We provide **causal evidence that the identified features support successive computation beyond the immediate prediction**: their removal disrupts later states and final answers, while restoring carry-related features to base values selectively impairs carry-dependent solutions.

2 METHOD

Our goal is to retain the parts of a fine-tuning update that support learned mathematical performance under a limited storage budget. Figure 1 summarizes our approach. A shared SAE identifies feature changes induced by fine-tuning, and these changes are combined with prediction loss to select update components. The selected components are applied to the base model without requiring the SAE at inference.

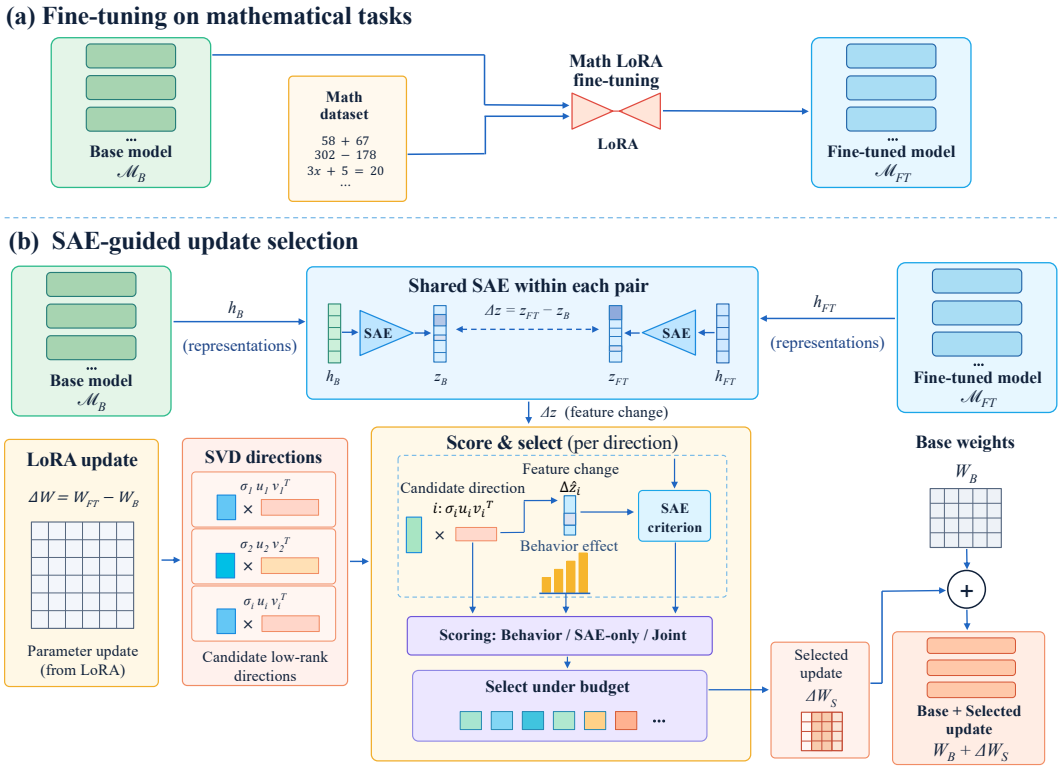


Figure 1: Overview of feature-guided parameter selection. (a) Fine-tuning on mathematical tasks produces a LoRA update. (b) A shared SAE identifies feature changes between the base and fine-tuned models. Singular components of the update are scored using prediction loss, feature matching, or both, and selected under a fixed factor-storage budget. Inference uses only the base model and the selected update.

2.1 Identifying Feature Changes

Let W_{B} and W_{M} denote the parameters before and after fine-tuning, respectively. We compare their layer-ℓ representations, $h^{\mathrm{B}}_{\ell,t}$ and $h^{\mathrm{M}}_{\ell,t}$, under identical token prefixes. As shown in the upper part of Figure 1(b), a shared SAE maps both representations into common feature coordinates, allowing us to measure how each feature changes after fine-tuning.

For each model pair and analyzed layer, we train an SAE with encoder $z_\ell(h) = \mathrm{TopK}_K(\mathrm{ReLU}(E_\ell h + b_\ell))$ and decoder $\widehat{h} = D_\ell z_\ell(h)$, where TopK_K retains the K largest activations. The training objective reconstructs both models' activations and their difference, with an additional term encouraging feature activity. We then freeze the SAE and rank features by $e_f = \sum_{q,v}(z_{\mathrm{M},qv,f} - z_{\mathrm{B},qv,f})^2$, where q indexes feature-selection problems and v their wording variants. These activations are measured at the position predicting the first computation token. The top k features define F. Appendix B details feature selection and scoring.

2.2 Scoring Update Components

The lower part of Figure 1(b) connects feature changes to parameter selection. We decompose the LoRA update into fixed rank-one components and score their effects on two objectives: predicting correct solutions and recovering the fine-tuned model's selected feature values.

Candidate components. For each adapted matrix j in a predefined layer window, singular value decomposition gives $\Delta W_j = U_j \Sigma_j V_j^\top$, including the LoRA scaling factor. We introduce coeffi-

cients g_{jk} to control the contribution of each component:

$$P_{jk} = \sigma_{jk} u_{jk} v_{jk}^{\top}, \qquad W_j(g) = W_{\mathrm{B},j} + \sum_{k=1}^{R_j} g_{jk} P_{jk}. \tag{1}$$

The components remain fixed throughout selection. Thus, $g = \mathbf{0}$ recovers the base model, while $g = \mathbf{1}$ applies the complete update within the candidate window. Parameters outside this window retain their base values.

Prediction and feature objectives. For a prompt x_q and its correct solution y_q, the prediction objective measures token-level negative log-likelihood. The feature objective measures normalized deviations from the fine-tuned model at annotated computation-token prediction positions:

$$\begin{aligned} \mathcal{L}_{\mathrm{beh}}(g) &= -\mathbb{E}_q \left[\frac{1}{L_q} \sum_{t=1}^{L_q} \log p_{W(g)}(y_{q,t} \mid x_q, y_{q,<t}) \right], \\ \mathcal{L}_F(g) &= \mathbb{E}_q \left[\frac{1}{|T_q||F|} \sum_{t \in T_q} \sum_{f \in F} \left(\frac{z_f(h_g(q,t)) - z_f(h_{\mathrm{M}}(q,t))}{v_f} \right)^2 \right]. \end{aligned} \tag{2}$$

Here, L_q includes EOS, and T_q contains the annotated positions. All models receive the same correct prefix at each position. The scale v_f is the root mean square of the feature's base–fine-tuned difference on feature-selection data, floored at $\epsilon > 0$. Gradients pass through the frozen SAE encoder. Relative to the shared base reference, matching these feature values also matches their changes after fine-tuning.

Component scores. A component's effect depends on the other components present in the model. We therefore evaluate its sensitivity at multiple points along the path from the base model to the updated model:

$$s^{\mathcal{L}} = -\frac{1}{M} \sum_{m=1}^{M} \nabla_g \mathcal{L}(g)\big|_{g=\alpha_m \mathbf{1}}, \qquad \alpha_m \in (0,1). \tag{3}$$

Each entry estimates the local loss reduction from increasing one component, averaged across update strengths. This provides a first-order selection criterion rather than an exact measure of the component's removal effect. Let $b = s^{\mathcal{L}_{\mathrm{beh}}}$ and $a = s^{\mathcal{L}_F}$. We define three scoring variants: ***Behavior*** uses the prediction-based score b, ***SAE-only*** uses the feature-based score a, and ***Joint*** combines them through norm matching:

$$s = b + \lambda_{\mathrm{feat}} \frac{\|b\|_2}{\|a\|_2} a. \tag{4}$$

Here, λ_{feat} controls the relative contribution of feature matching. Appendix B details scoring.

2.3 Selecting under a Storage Budget

The final stage in Figure 1(b) converts component scores into a compact update. Storing a rank-one component of matrix j requires $c_j = d_{\mathrm{in},j} + d_{\mathrm{out},j}$ factor scalars. To compare scoring objectives at equal cost, we group identically shaped matrices into sets $\mathcal{B}$ and assign each group the storage budget of a uniform rank-r adapter. Selection maximizes a linear surrogate, the sum of retained scores:

$$\begin{aligned} \max_{\{m_{jk} \in \{0,1\}\}} \quad & \sum_{j,k} m_{jk} s_{jk} \\ \text{subject to} \quad & \sum_{j \in \mathcal{B}} \sum_{k=1}^{R_j} m_{jk} = r|\mathcal{B}|, \qquad \text{for each group } \mathcal{B}. \end{aligned} \tag{5}$$

Because components within each group have equal storage cost, this objective is solved by retaining its $r|\mathcal{B}|$ highest-scoring components. The allocation permits different ranks across matrices while preserving the total factor-storage cost of uniform rank r. The SVD baseline instead retains the r largest singular components of each matrix.

For the selected indices $\mathcal{S}_j = \{k : m_{jk} = 1\}$, we construct $\Delta W_{\mathcal{S},j} = \sum_{k \in \mathcal{S}_j} P_{jk}$ and use $\widetilde{W}_j = W_{\mathrm{B},j} + \Delta W_{\mathcal{S},j}$ for generation. Selected components retain their original singular values and are stored as low-rank factors. The resulting model requires neither the SAE nor access to the fine-tuned model at inference. Appendix C details the storage calculation and factor construction.

3 Experiments and Analysis

3.1 Tasks and Data

We study three mathematical tasks—column addition, column subtraction, and variable isolation—with automatically verified solutions and intermediate states. The state annotations capture carry and borrow transitions in arithmetic and successive transformations in equations, enabling evaluation of intervention effects throughout a solution. Arithmetic training problems contain two- to six-digit operands, ordered from larger to smaller. Equations are generated from integer coefficients and nonzero integer solutions in three forms: $ax + b = c$, $a(x + b) = c$, and $ax + b = dx + c$. The fine-tuning set contains 21,000 problems: 7,500 additions, 7,500 subtractions, and 6,000 equations. For feature selection, we use 384 problems, each rendered in three natural-language formulations and one symbolic form. Parameter scoring uses a subset of 96 problems with the same formulations. Splits are assigned by underlying mathematical content, keeping related problems and all wording variants together. Appendix A details data generation, state annotations, and split construction.

3.2 Experimental Setup

Models. We fine-tune Qwen2.5-1.5B-Base, Qwen3-4B-Base, Gemma-3-4B-PT, and Llama-3-8B-Base on the 21,000 problems in Section 3.1, using rank-64 LoRA and seeds 43, 44, and 45. Base and Full LoRA denote the original and fully fine-tuned models.

Selection methods and budgets. SVD, Behavior, SAE-only, Joint, and Random use the same candidate matrices and storage budget within each pair. SVD retains the largest singular components within each matrix. Behavior, SAE-only, and Joint use the scores in Section 2.2. Random selects components at random under the same allocation constraints. Equivalent ranks $r \in \{4, 8, 16, 32\}$ specify the factor-storage cost of uniform rank r. Our main setting, $r = 16$, uses 25% of the full rank-64 adapter's factor storage. Scored selection allocates this budget across matrices of the same shape.

Evaluation protocol. In-distribution (ID) evaluation covers the fine-tuning tasks, while GSM8K (Cobbe et al., 2021) evaluates performance beyond these tasks. Interventions begin from common correct prefixes and measure current-token, subsequent-state, and final-answer accuracy. Free continuation predicts the next token, whereas fixed-token continuation supplies the correct one before resuming generation. Appendix E defines these evaluation conditions.

Parameter-selection results average seeds 43–45. Feature discovery, scoring, and hyperparameter selection precede held-out intervention evaluation. Protocol details are provided in Appendices A, B, and E, with full results reported in Appendix F.

3.3 Feature Changes Guide Effective Parameter Selection

We begin by comparing selection objectives at equal storage cost. We then vary the budget to test whether the comparison holds across update sizes and remove selected parameter groups to examine their contributions to each task.

Feature matching improves on prediction-loss scoring. To address which parameter changes preserve the acquired capability, we test whether fine-tuning-induced feature changes provide a useful signal for selecting update components beyond prediction loss alone. At the main budget in Figure 2, Behavior exceeds SVD by 2.9–12.1 percentage points. Feature matching provides a further gain: SAE-only exceeds Behavior by 5.2–13.5 points across all four pairs. Behavior, SAE-only, and Joint share the same candidate components, budget, and allocation rule, isolating the scoring objective in this comparison. Joint improves over Behavior by 8.2–12.8 points and achieves

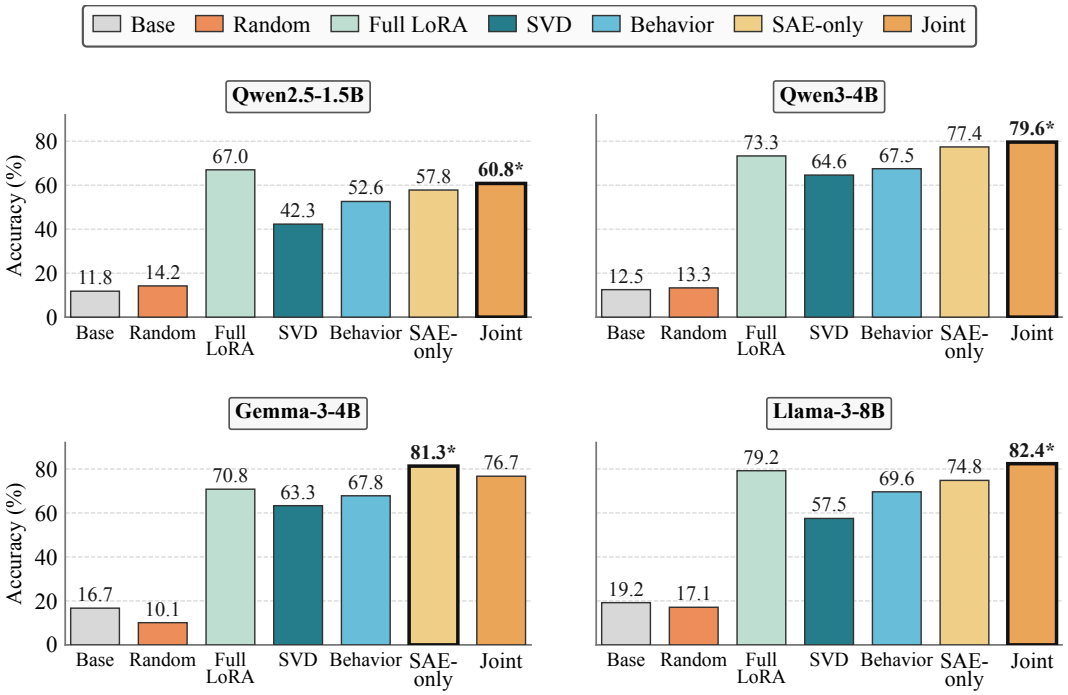


Figure 2: ID answer accuracy at 25% of the full adapter's factor storage, averaged over three fine-tuning seeds. Asterisks mark the best selection method for each model. Base and Full LoRA provide reference accuracies.

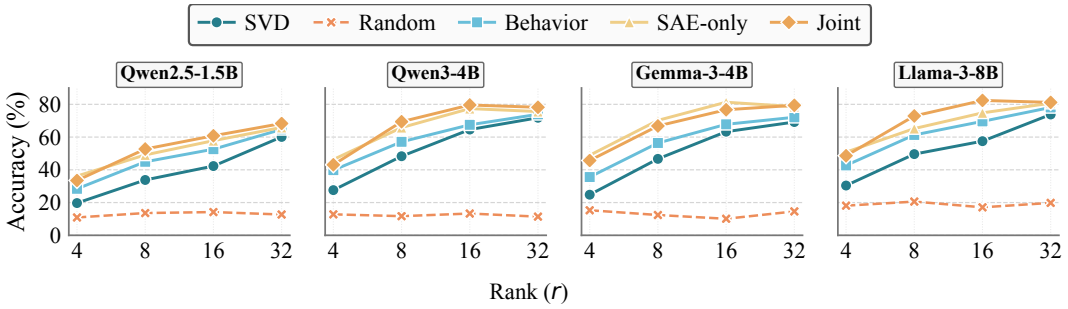


Figure 3: ID answer accuracy across equivalent ranks. Ranks 4, 8, 16, and 32 correspond to 6.25%, 12.5%, 25%, and 50% of rank-64 factor storage. SAE-only and Joint outperform Behavior and SVD throughout the displayed budgets.

the best accuracy on three model pairs, whereas SAE-only performs best on Gemma-3-4B at 81.3%, compared with 76.7% for Joint.

A compact update preserves substantial fine-tuning gains. Comparison with the complete update shows how much performance the selected components retain. Joint exceeds Full LoRA by 6.3, 5.9, and 3.2 points on Qwen3-4B, Gemma-3-4B, and Llama-3-8B, respectively. On Qwen2.5-1.5B, it raises accuracy from Base's 11.8% to 60.8%, compared with 67.0% for Full LoRA. Random selection remains near or below Base across the four pairs. The retained components therefore matter: feature guidance preserves substantial fine-tuning gains with one quarter of the factor storage and surpasses the complete update on three pairs.

Feature guidance remains effective across budgets. Figure 3 shows that SAE-only and Joint outperform Behavior and SVD at every displayed budget. Accuracy generally improves as more components are retained, although some curves peak at rank 16. The relative ordering of SAE-only

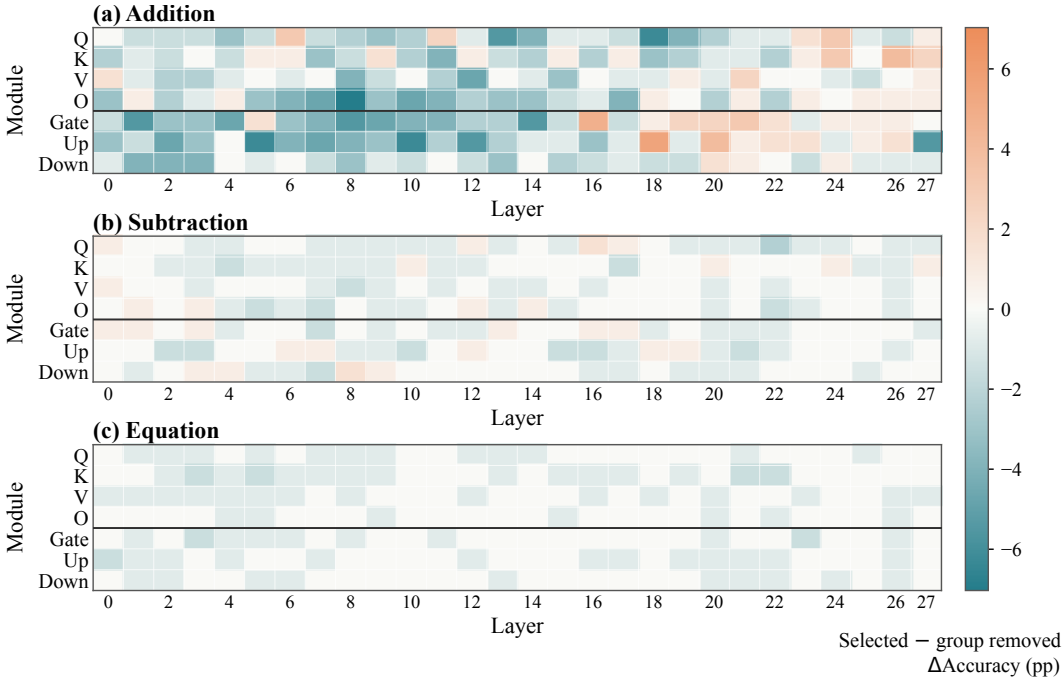


Figure 4: Task-specific effects of removing selected parameter groups in Qwen2.5-1.5B. Values are answer accuracy before removal minus accuracy after removal, in percentage points. Positive values indicate supporting groups, while negative values indicate that removal improves performance. Horizontal lines separate attention and MLP projections.

and Joint varies across models and budgets. Their common advantage over Behavior shows that feature guidance remains useful from the smallest budget onward.

Parameter contributions depend on the task. To examine how performance depends on different parts of the selected update, we remove the selected components of each layer–projection group in Qwen2.5-1.5B while retaining the rest of the update. Figure 4 shows that addition is most sensitive, particularly to MLP groups, whereas subtraction and variable isolation are less affected. Addition also exhibits effects of both signs: several later-layer groups support accuracy, while removing others improves it. The selected groups thus make distinct contributions across tasks, with each effect measured in the context of the remaining update.

3.4 Feature Contributions Extend beyond the Next Token

The selection results establish the value of feature guidance for task performance. We next examine what the features contribute during a solution. Comparing free continuation with continuation after supplying the correct next token tests whether their effects on later states and answers persist once the immediate prediction is correct.

Interventions and controls. Under the same correct prefix, we replace the Base residual with the Full LoRA residual, insert selected features into Base, or delete them from Full LoRA. Insertion tests whether the features support Base computation, deletion tests whether Full LoRA relies on them. Feature interventions add or subtract decoded SAE contributions. Controls use random features and norm-matched residual noise (Appendix D). Table 1 reports free and fixed-token continuation.

Improving the immediate prediction does not ensure a correct solution. Full residual replacement raises Base's token accuracy from 59.52% to 79.37%, but final-answer accuracy falls from 28.51% to 23.81%. Selected-feature insertion produces a smaller token improvement, to 73.81%,

Table 1: Feature interventions under free and fixed-token continuation. Token, State, and Answer report current-token, subsequent-state, and final-answer accuracy (%). Fixed-token continuation supplies the correct next token in every condition.

	Free continuation			Fixed-token continuation	
Intervention	**Token**	**State**	**Answer**	**State**	**Answer**
Panel A: Base model					
No intervention	59.52	4.76	28.51	7.35	26.19
Full residual replacement	**79.37**	9.52	**23.81**	9.52	23.81
Selected SAE feature insertion	73.81	7.14	**30.95**	7.14	**31.57**
Random SAE feature insertion	58.19	4.62	24.14	6.47	22.78
Norm-matched random residual noise	61.90	3.88	26.19	4.46	24.72
Panel B: Full LoRA model					
No intervention	77.62	88.10	97.62	90.48	97.61
Selected SAE feature deletion	62.86	**73.13**	**75.24**	**78.10**	**72.89**
Random SAE feature deletion	72.24	84.91	90.76	86.14	85.52

yet raises answer accuracy to 30.95%. Its answer accuracy also exceeds both random controls. These interventions separate success on the immediate prediction from success on the remaining solution.

With the next token fixed, selected-feature insertion reaches 31.57% answer accuracy, versus 26.19% without intervention and 22.78% with random-feature insertion. The continuation benefit therefore persists when all conditions receive the same correct next token.

Feature deletion disrupts subsequent computation. Deleting selected features from Full LoRA lowers free-continuation state accuracy from 88.10% to 73.13% and answer accuracy from 97.62% to 75.24%, exceeding the losses from random-feature deletion. Fixing the next token does not remove the effect: state and answer accuracy fall from 90.48% and 97.61% to 78.10% and 72.89%, respectively. The selected features therefore affect subsequent computation even when the immediate prediction is corrected.

3.5 Retained Feature Changes Support Carry Computation

We next test whether the retained feature changes support a specific computation: carrying a value from one column to the next. We examine eight carry-related features whose identities are fixed before evaluation. Full LoRA and Joint exhibit similar response patterns. For example, Feature 731 has mean activations of 2.123 without carry and 0.716 with carry in Joint, indicating that the retained update preserves carry-sensitive feature responses.

To test whether these responses contribute causally to carry computation, we restore the eight features in Joint to their Base values under matched prefixes while leaving the selected parameter update unchanged (Appendix D.4). On 2,800 carry problems, restoration reduces current-carry, subsequent-state, and final-answer accuracy by 28.1, 15.5, and 26.9 percentage points, respectively, and lowers answer accuracy by 22.1 points relative to random-feature restoration. Effects on 2,800 matched no-carry controls remain small. These results indicate that the retained feature changes specifically support carry-dependent computation.

3.6 Preserving Mathematical Performance beyond Fine-Tuning

The preceding analyses show that selected updates retain learned task performance and features supporting computation. We now ask whether these gains come at the expense of mathematical performance beyond the fine-tuning tasks. Table 2 shows that Full LoRA reduces GSM8K accuracy relative to Base on all four model pairs, revealing a trade-off between specialization and retention.

Joint partially mitigates the specialization–retention trade-off. On three of four model pairs, it improves both ID and GSM8K accuracy over Full LoRA, showing that retaining the complete update

Table 2: GSM8K answer accuracy (%) for the same models evaluated on ID tasks. Selection methods use 25% of the full adapter's factor storage.

Method	Qwen2.5-1.5B	Qwen3-4B	Gemma-3-4B	Llama-3-8B
Base	32.1	37.5	37.3	49.9
Full LoRA	30.7	34.2	25.8	43.4
SVD	27.9	30.8	28.7	44.8
Random	29.5	36.4	34.2	49.0
Behavior	24.6	29.7	35.9	45.4
SAE-only	26.2	33.8	34.0	46.2
Joint	27.4	**35.8**	**36.2**	**47.6**

is not necessary to preserve either newly acquired performance or broader mathematical capability. The strongest recovery occurs on Gemma-3-4B, where Joint restores 10.4 percentage points of GSM8K accuracy. However, the effect remains model-dependent, and GSM8K performance does not fully return to Base.

Importantly, retention alone does not imply successful task learning. Random selection can preserve GSM8K accuracy while failing to retain the fine-tuned tasks, whereas Joint achieves substantial gains on both. These results suggest that feature-guided selection improves the balance between specialization and retention rather than optimizing either in isolation.

4 Related Work

Mechanistic interpretability and model changes. Prior work studies how internal representations and parameter changes support model behavior. Activation patching probes causal effects of representations (Zhang & Nanda, 2024), while SAEs provide sparse feature coordinates (Huben et al., 2024; Gao et al., 2025). Feature circuits and crosscoder-based model diffing further analyze causal structure and model differences (Marks et al., 2025; Mishra-Sharma et al., 2025). At the parameter level, task arithmetic operates on weight differences (Ilharco et al., 2022), and Task Singular Vectors decompose such updates into singular directions (Gargiulo et al., 2025). Related mechanistic work also connects editing behavior to singular parameter directions (Xie et al., 2025).

Mechanisms of mathematical reasoning. Mechanistic studies have identified representations and circuits underlying modular addition, numerical comparison, and arithmetic behavior (Nanda et al., 2023; Zhong et al., 2023; Hanna et al., 2023; Nikankin et al., 2025). Recent work also transfers mathematical reasoning abilities across models through parameter-space manipulation (Hu et al., 2026). Our work connects these directions by using shared SAE features to guide post-hoc selection of fine-tuning updates and test whether the associated representations support successive mathematical computation.

5 Conclusion

We study which fine-tuning changes retain acquired mathematical capabilities and how the associated representations contribute to computation. We introduce **SAE-guided update selection**, which uses shared sparse features to identify effective low-rank update components. Across four model pairs, feature guidance consistently improves selection, and Joint exceeds Full LoRA on both target-task and GSM8K accuracy for three pairs using only 25% of the adapter's factor storage. These results suggest that acquired mathematical performance can be retained by a compact subset of the fine-tuning update. Feature interventions further show that the identified representations **support computation beyond the immediate prediction**: their removal disrupts subsequent states and final answers even after the correct next token is supplied. Carry-specific restoration further links these features to a concrete computational role by selectively impairing carry-dependent solutions. Together, our results connect **parameter changes, representation changes, and successive computation** in compact mathematical updates.

Reproducibility Statement

Section 2 defines the selection objectives and budget. Appendices A–C provide data construction, feature selection, component scoring, and factor export. Appendices D–F detail interventions, evaluation conventions, and supplementary numerical results. Anonymous source code will be made available to reviewers and area chairs during the discussion period, and the full codebase will be publicly released after decision notification.

AI Use Statement

Generative AI tools assisted with manuscript organization, language editing, and the presentation and interpretation of experimental results, as well as with generating the synthetic datasets used in this study. Section 3.1 and Appendix A describe data construction and verification. The authors are responsible for reviewing and verifying all AI-assisted content and generated data, and take full responsibility for the final manuscript and reported results.

References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Epoch AI. Capabilities & benchmarking, 9 2026. URL `https://epoch.ai/benchmarks/use-this-data`. Accessed: 25 Sep 2026.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations*, volume 2025, pp. 26721–26754, 2025.

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodola. Task singular vectors: Reducing task interference in model merging. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18695–18705. IEEE, 2025.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Yijie Hu, Zihao Zhou, Kaizhu Huang, Xiaowei Huang, and Qiufeng Wang. Can mllms absorb math reasoning abilities from llms as free lunch? volume 38, pp. 130912–130940, 2026.

Robert Huben, Hoagy Cunningham, Logan Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations*, volume 2024, pp. 7827–7845, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Samuel Marks, Can Rager, Eric Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations*, volume 2025, pp. 23888–23923, 2025.

Siddharth Mishra-Sharma, Trenton Bricken, Jack Lindsey, Adam Jermyn, Jonathan Marcus, Kelley Rivoire, Christopher Olah, and Thomas Henighan. Insights on crosscoder model diffing. Transformer Circuits Thread, 2025. URL `https://transformer-circuits.pub/2025/crosscoder-diffing-update/index.html`.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *International Conference on Learning Representations*, volume 2025, pp. 55939–55965, 2025.

OpenAI. GPT-6 Astra: A New Generation of Intelligence, 2026. URL `https://openai.com/index/gpt-6-astra/`. Accessed: September 25, 2026.

Jiakuan Xie, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Revealing the deceptiveness of knowledge editing: A mechanistic analysis of superficial editing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17756–17780, 2025.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, volume 2024, pp. 1651–1678, 2024.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. *Advances in neural information processing systems*, 36:27223–27250, 2023.

A Data Construction and Computational States

Each problem has a verified sequence of computational states shared by its wording variants. The following rules define the reference states and final answers used in evaluation.

A.1 Column Addition and Subtraction

Let a, b be nonnegative integers with $a \geq b$, and let a_i, b_i be their base-ten digits, indexed from the least significant column $i = 0$. Training uses operands with two to six digits. For addition, the incoming carry is $c_0 = 0$, and each column follows

$$t_i = a_i + b_i + c_i, \qquad y_i = t_i \bmod 10, \qquad c_{i+1} = \lfloor t_i/10 \rfloor . \tag{6}$$

The column state records i, a_i, b_i, c_i, y_i, c_{i+1}, and the accumulated output digits. A final carry supplies the most significant digit; place values determine the completed answer.

For subtraction, c_i denotes the incoming borrow and $c_0 = 0$. The transition is

$$t_i = a_i - b_i - c_i, \qquad c_{i+1} = \mathbf{1}[t_i < 0], \qquad y_i = t_i + 10c_{i+1}. \tag{7}$$

The ordering $a \geq b$ makes the result nonnegative. These equations determine both the reference answer and each carry or borrow transition. Integer arithmetic verifies the output digits and their accumulated value.

For example, the units column of $47 + 28$ gives $7 + 8 + 0 = 15$, so its output digit is 5 and its outgoing carry is 1. The tens column then gives $4 + 2 + 1 = 7$. The answer is 75. In $52 - 37$, the units column produces digit 5 and borrow 1; the tens column uses that borrow to produce digit 1. These examples illustrate how an intermediate state supplies an input to the next computation step.

A.2 Variable Isolation

The equation tasks use three forms. In $ax + b = c$, with $a \neq 0$, the transformations give $ax = c - b$ and $x = (c - b)/a$. In $a(x + b) = c$, division followed by subtraction gives $x + b = c/a$ and $x = c/a - b$. For $ax + b = dx + c$, with $a \neq d$, collecting terms gives $(a - d)x + b = c$, then $(a - d)x = c - b$, and finally $x = (c - b)/(a - d)$.

Construction begins with integer coefficients and a nonzero integer solution x. The constant on the right is computed so that the equation holds: $c = ax + b$, $c = a(x + b)$, or $c = (a - d)x + b$, respectively. This gives a known solution for each generated problem. SymPy checks the solution and the recorded transformations.

A.3 Problem Groups and Data Allocation

Fine-tuning uses 21,000 problems, comprising 7,500 additions, 7,500 subtractions, and 6,000 equations. Feature selection uses 384 problems, and parameter scoring uses a subset of 96. Each problem in these two stages has four wordings: three natural-language formulations and one symbolic formulation. These stages therefore contain 1,536 and 384 inputs, respectively.

Deduplication operates on mathematical content. Arithmetic problems share a key when they have the same unordered operand pair, including across addition and subtraction. Equation keys use the equation form and its generating parameters. Related problems and wording variants are assigned together. This grouping also defines the unit used for paired comparisons across wordings.

Table 3: Synthetic-data allocations. Counts refer to mathematical problems unless labeled as wording instances. Parameter scoring uses a subset of feature-discovery problems.

Use	Problems	Composition or rendering
Fine-tuning	21,000	7,500 addition; 7,500 subtraction; 6,000 equations
Feature discovery	384	Four wordings each; 1,536 wording instances
Parameter scoring	96	Four wordings each; 384 wording instances
Carry evaluation	2,800	Addition problems requiring carry
No-carry evaluation	2,800	Addition problems without carry

Parameter scoring uses 96 of the 384 feature-discovery problems. Related mathematical problems and their wording variants are assigned to the same split.

A.4 Examples from the Training Data

Table 4 illustrates the stored training problems and their computational states. The examples cover different carry and borrow patterns and all three equation forms; the descriptive arithmetic categories can overlap.

Table 4: Training examples in compact mathematical notation. Arithmetic states list $(\text{output digit}, \text{outgoing carry/borrow})$ from the least significant column onward. Equation states retain the intermediate equalities; the final solution appears in the Answer column.

Pattern	Problem	Computational states	Answer
Column addition			
No carry	$71 + 12$	$(3,0),\ (8,0)$	83
Single carry	$77 + 15$	$(2,1),\ (9,0)$	92
Consecutive carry	$497 + 144$	$(1,1),\ (4,1),\ (6,0)$	641
Final carry	$910 + 740$	$(0,0),\ (5,0),\ (6,1);\ 1^{\dagger}$	1650
Column subtraction			
No borrow	$69 - 61$	$(8,0),\ (0,0)$	8
Single borrow	$722 - 160$	$(2,0),\ (6,1),\ (5,0)$	562
Consecutive borrow	$744 - 466$	$(8,1),\ (7,1),\ (2,0)$	278
Borrow across zero	$704 - 599$	$(5,1),\ (0,1),\ (1,0)$	105
Variable isolation			
$ax + b = c$	$8x + 4 = 52$	$8x = 48$	$x = 6$
$a(x + b) = c$	$-6(x - 69) = 492$	$x - 69 = -82$	$x = -13$
$ax + b = dx + c$	$2x + 48 = 3x + 6$	$-x + 48 = 6 \ \rightarrow\ -x = -42$	$x = 42$
$ax + b = dx + c$	$7x + 1 = 8x + 7$	$-x + 1 = 7 \ \rightarrow\ -x = 6$	$x = -6$

[†]The stored `emit_final_carry` operation appends the leading digit 1. Full arithmetic records also contain the incoming carry/borrow, operand digits, column position, and accumulated output digits; these fields are omitted here for space.

The stored targets serialize state transitions as `s_i: {state} -> operation -> s_j: {state}`, followed by an `Answer:` field. For example, the selected prompt for $497 + 144$ is

`497+144=?; states`, followed by a newline and `Solution:`. Training appends EOS to the target. Although each source problem stores four prompt variants, only one is selected per problem for fine-tuning.

B Feature Selection and Parameter Scoring

B.1 Shared SAE Coordinates

At each analyzed layer, a shared encoder and decoder process Base and Full LoRA activations under identical token prefixes. A feature is indexed by the layer–coordinate pair (ℓ, f). Each layer and model pair has its own SAE. Layer indices are suppressed below when the layer is fixed.

For hidden dimension d and dictionary width p, the encoder and decoder in Section 2.1 have dimensions $E \in \mathbb{R}^{p\times d}$ and $D \in \mathbb{R}^{d\times p}$. The encoder retains at most K nonzero coordinates for each input. This encoder sparsity K differs from the number k of coordinates selected for parameter scoring: K controls an individual activation vector, whereas k controls a fixed set of feature identities.

SAE training reconstructs Base activations, Full LoRA activations, and their difference, with an additional term encouraging feature activity. After training, the encoder and decoder remain fixed. The role of the difference reconstruction can be expressed by the identity

$$(\widehat{h}^{\mathrm{M}} - \widehat{h}^{\mathrm{B}}) - (h^{\mathrm{M}} - h^{\mathrm{B}}) = (\widehat{h}^{\mathrm{M}} - h^{\mathrm{M}}) - (\widehat{h}^{\mathrm{B}} - h^{\mathrm{B}}). \tag{8}$$

Thus, reconstructing the difference constrains the discrepancy between the two reconstruction errors.

B.2 Feature Discovery and Normalization

Discovery measures feature changes at the position predicting the first computation token. Let $\mathcal{D}_{\mathrm{disc}}$ contain the discovery problem–wording pairs. The ranking statistic is

$$e_f = \sum_{(q,v)\in\mathcal{D}_{\mathrm{disc}}} \left(z_{\mathrm{M},qv,f} - z_{\mathrm{B},qv,f}\right)^2. \tag{9}$$

The largest squared changes define the selected feature set F. This statistic counts increases and decreases alike. Once selected, the feature identities remain fixed during scoring.

For a selected coordinate, let $\mathcal{I}_{\mathrm{disc},f}$ denote the discovery activation pairs used to estimate its scale. The normalization in Equation 2 is

$$v_f = \max\left\{\left[\frac{1}{|\mathcal{I}_{\mathrm{disc},f}|}\sum_{i\in\mathcal{I}_{\mathrm{disc},f}}\left(z_{\mathrm{M},i,f} - z_{\mathrm{B},i,f}\right)^2\right]^{1/2}, \epsilon\right\}, \qquad \epsilon > 0. \tag{10}$$

This is an uncentered RMS difference. Both the scales and the Full-LoRA targets are fixed while differentiating the scoring loss. The normalization measures matching error relative to each feature's observed change.

Discovery and scoring use different position sets. Discovery ranks coordinates at the first computation-token prediction position; scoring matches them at all annotated computation-token prediction positions T_q. At a target token $y_{q,t}$, the relevant representation is computed from $(x_q, y_{q,<t})$, before that target is supplied. This distinction prevents the target token from being included in its own prediction prefix.

The Random baseline selects parameter components. Random-feature insertion, deletion, and restoration intervene on representation coordinates (Appendix D).

B.3 Scoring and Selection Algorithm

Write $\mathcal{J}$ for the candidate matrices and $N = \sum_{j\in\mathcal{J}} R_j$ for the number of candidate components. Each $P_{jk} = \sigma_{jk} u_{jk} v_{jk}^{\top}$ includes the original LoRA scale and remains fixed. Only its scalar coefficient g_{jk} is differentiated.

Loss masks and averaging. The behavior loss averages the NLL of the complete correct solution, including EOS, within each example, then averages across examples. The behavior-loss mask covers solution tokens. The feature loss first averages squared normalized deviations over $T_q \times F$, then averages across examples. These per-example means give each example equal weight irrespective of solution length. The feature mask follows annotated computation positions. Gradients propagate through the frozen encoder to the model and then to g.

Table 5: SAE-guided selection with fixed candidate components. All objectives refer to Equation 2. The Joint expression assumes $\|a\|_2 > 0$.

Input: Base weights; scaled LoRA updates on $\mathcal{J}$; frozen SAEs; F and v_f; scoring examples; shape groups; equivalent rank r; coefficient λ_{feat}.

1. Decompose each update into fixed P_{jk} and initialize $a, b \leftarrow 0 \in \mathbb{R}^N$.
2. Fix Full-LoRA feature targets under the correct solution prefixes.
3. For each $\alpha \in \{0.25, 0.5, 0.75\}$, set $g \leftarrow \alpha \mathbf{1}$. Evaluate the two losses with the same correct prefixes, and accumulate
$$b \leftarrow b - \tfrac{1}{3}\nabla_g \mathcal{L}_{\text{beh}}(g), \qquad a \leftarrow a - \tfrac{1}{3}\nabla_g \mathcal{L}_F(g).$$
4. Set $s = b$ for Behavior, $s = a$ for SAE-only, or $s = b + \lambda_{\text{feat}}(\|b\|_2/\|a\|_2)a$ for Joint.
5. In each shape group $\mathcal{B}$, retain the $r|\mathcal{B}|$ components with largest s_{jk}. Let $\mathcal{S}_j$ contain the selected indices of matrix j.
6. Export $\sum_{k\in\mathcal{S}_j} P_{jk}$ using the factors in Appendix C.2. Retained coefficients equal one; omitted coefficients equal zero.

Output: Base weights with the selected low-rank update.

The norms in Joint range over the complete candidate score vectors, before shape-group selection. For nonzero a and $\lambda_{\text{feat}} \geq 0$,

$$\left\| \lambda_{\text{feat}} \frac{\|b\|_2}{\|a\|_2} a \right\|_2 = \lambda_{\text{feat}} \|b\|_2. \tag{11}$$

This fixes the relative norm of the feature contribution. The interpolation coefficients are used to score components; retained components are exported with coefficient one.

B.4 First-Order Interpretation and Interactions

Let e_{jk} be the unit vector for one coefficient. At a differentiable point g,

$$\mathcal{L}(g + \delta e_{jk}) - \mathcal{L}(g) = \delta \frac{\partial \mathcal{L}(g)}{\partial g_{jk}} + o(\delta). \tag{12}$$

The negative derivative measures local loss reduction from increasing that component. By the chain rule, it is also a directional derivative in weight space:

$$\frac{\partial \mathcal{L}(g)}{\partial g_{jk}} = \left\langle \nabla_{W_j} \mathcal{L}(g), P_{jk} \right\rangle_F, \tag{13}$$

where $\langle A, B\rangle_F = \operatorname{tr}(A^\top B)$. The score averages these sensitivities at three points on the uniform interpolation path.

The budgeted objective is linear in the retained scores. Where a second-order expansion exists, a simultaneous change d satisfies

$$\mathcal{L}(g + d) - \mathcal{L}(g) = \nabla_g \mathcal{L}(g)^\top d + \tfrac{1}{2} d^\top H(g) d + o(\|d\|_2^2), \tag{14}$$

where $H(g)$ is the Hessian in coefficient space. Off-diagonal terms capture component interactions omitted by the linear surrogate. The effect of finite deletion depends on the other retained components. Parameter-group removal in Appendix D.3 measures this conditional effect.

C Storage Budget and Factor Construction

C.1 Equal-Cost Allocation

An adapted matrix has shape $d_{\text{out},j} \times d_{\text{in},j}$. A rank-one component stored as two vectors requires $c_j = d_{\text{out},j} + d_{\text{in},j}$ scalars, with its singular value absorbed into the factors. If matrix j retains r_j components, its group and total costs satisfy

$$C_{\mathcal{B}} = c_{\mathcal{B}} \sum_{j \in \mathcal{B}} r_j = r|\mathcal{B}|c_{\mathcal{B}}, \qquad C(r) = \sum_{\mathcal{B}} C_{\mathcal{B}} = r \sum_{j \in \mathcal{J}} c_j. \tag{15}$$

The equality follows from the group constraint $\sum_{j \in \mathcal{B}} r_j = r|\mathcal{B}|$. It requires enough candidate components to fill each group budget.

Within a group, exchanging a selected lower-scoring component for an unselected higher-scoring component increases the objective at unchanged cost. Repeating this exchange proves that retaining the highest $r|\mathcal{B}|$ scores solves Equation 5. The equality constraint fills the budget even if some retained scores are negative. A tie at the cutoff can produce multiple optimal selections.

SVD uses a uniform rank in each matrix, whereas scored methods allow unequal ranks within a shape group. In particular, the Behavior–SAE-only–Joint comparison holds the allocation rule fixed, while the comparison with SVD changes both the scoring criterion and the allocation rule.

Equivalent ranks $r \in \{4, 8, 16, 32\}$ use 6.25%, 12.5%, 25%, and 50% of rank-64 factor storage over the same candidate matrices, respectively.

C.2 Candidate Window and Exported Factors

Only matrices in $\mathcal{J}$ receive candidate updates; parameters outside this window retain their Base values. Thus, $g = \mathbf{1}$ applies the complete candidate-window update. It equals Full LoRA only when that window includes all nonzero adapted updates. The storage ratio $C(r)/C(64)$ uses the same candidate matrices in the numerator and denominator.

For selected components $\mathcal{S}_j$, define

$$\widetilde{B}_j = \left[\sqrt{\sigma_{jk}} u_{jk}\right]_{k \in \mathcal{S}_j}, \qquad \widetilde{A}_j = \left[\sqrt{\sigma_{jk}} v_{jk}^\top\right]_{k \in \mathcal{S}_j}. \tag{16}$$

The first expression stacks columns, giving $d_{\text{out},j} \times r_j$ factors; the second stacks rows, giving $r_j \times d_{\text{in},j}$ factors. Their product is

$$\widetilde{B}_j \widetilde{A}_j = \sum_{k \in \mathcal{S}_j} \sigma_{jk} u_{jk} v_{jk}^\top. \tag{17}$$

No additional LoRA scale is applied to this product, because the scale is already included in the decomposed update. If no component is selected for a matrix, its exported update is zero. All exported factors remain fixed during evaluation.

D Interventions on Representations and Parameters

D.1 Residual and Feature Interventions

All paired activation comparisons use identical prefix tokens. Full residual replacement sets $h'_{\ell,t} = h^{\text{M}}_{\ell,t}$ in a Base forward pass at the intervention location. Feature interventions add a decoded change to the host residual:

$$h' = h + D\Delta z, \qquad \Delta z_f = 0 \quad \text{for } f \notin S, \tag{18}$$

where S is the intervened feature set. Selected-feature insertion adds decoded feature contributions to Base; deletion subtracts them from Full LoRA.

For the additive operation in Equation 18, writing the host residual as $h = Dz(h) + \rho(h)$ gives

$$h' = D\big(z(h) + \Delta z\big) + \rho(h). \tag{19}$$

Thus the existing reconstruction residual is preserved. These equations use the residual-coordinate decoder convention of Section 2.1.

Let Π_S be the diagonal coordinate mask for S. Joint-to-Base restoration uses

$$\Delta z = \Pi_S \left(z(h^{\mathrm{B}}) - z(h^{\mathrm{Joint}}) \right), \qquad h^{\mathrm{Joint}\to\mathrm{B}} = h^{\mathrm{Joint}} + D\Delta z. \tag{20}$$

Joint's selected parameter update remains in place. This operation adds the decoded coordinate difference.

Random-feature interventions change the coordinate set, whereas norm-matched residual noise controls perturbation magnitude in residual space.

D.2 Free and Fixed-Token Continuation

Both conditions start from a common correct prefix and apply the specified intervention at the prediction position. In free continuation, the model predicts the next token and generates the remaining solution. Current-token accuracy evaluates that immediate prediction; subsequent-state and answer accuracy evaluate the continuation.

In fixed-token continuation, the correct next token is supplied before generation resumes, including in the no-intervention condition. Thus the compared continuations share a correct immediate token. Their state and answer differences test whether the intervention affects later computation after that token is corrected.

D.3 Parameter-Group Removal

The group-removal analysis uses Qwen2.5-1.5B. Let G identify one layer and projection matrix, and let m denote the selected component mask. Define $m^{\setminus G}$ by zeroing its entries in G while leaving the others unchanged. For each task, the effect is

$$\Delta_G = \mathrm{Acc}(W(m)) - \mathrm{Acc}(W(m^{\setminus G})). \tag{21}$$

Positive values indicate a supporting group; negative values mean removal improves accuracy. Because all other selected updates remain in place, Δ_G is conditional on their presence.

D.4 Carry-Feature Restoration

The intervention uses a fixed set of eight carry-related features, denoted S_{carry}.

Restoration applies Equation 20 with $S = S_{\mathrm{carry}}$ under matched prefixes. Evaluation uses 2,800 carry problems and 2,800 no-carry problems. Table 6 summarizes the accuracy differences.

Table 6: Carry-restoration contrasts in percentage points. "Drop" subtracts selected restoration from no intervention; "random minus selected" compares the two restoration conditions.

Evaluation group	Metric and contrast	Difference
Carry	Current-carry accuracy: drop	28.1
Carry	Subsequent-state accuracy: drop	15.5
Carry	Answer accuracy: drop	26.9
Carry	Answer accuracy: random minus selected	22.1
No carry	Current-carry accuracy: absolute change	1.7
No carry	Answer accuracy: absolute random–selected difference	1.2

The absolute change in current-carry accuracy is larger on carry problems than on no-carry problems.

E Evaluation Conventions

E.1 Prediction Conditions and Metrics

ID and GSM8K evaluation use greedy generation. Answer accuracy is the proportion of outputs whose extracted final answer matches the verified reference. Parameter-selection evaluation gener-

ates complete solutions, whereas intervention evaluation begins from correct prefixes. Each therefore uses its own Base or no-intervention reference. Free and fixed-token continuation also have distinct references because the supplied information differs.

Current-token accuracy evaluates the prediction immediately following the supplied prefix. Subsequent-state accuracy compares later annotated computation states with the reference trajectory. Arithmetic states include digits, carry or borrow, and the accumulated result; equation states describe successive transformations. These metrics evaluate different units and are reported separately.

E.2 Seed Averaging

The main parameter-selection results average fine-tuning seeds $\mathcal{S} = \{43, 44, 45\}$. For method m, the mean accuracy is $\bar{A}_m = \frac{1}{3} \sum_{s \in \mathcal{S}} A_{m,s}$, where $A_{m,s}$ is its accuracy in seed s.

Percentage-point contrasts subtract accuracies expressed on the percentage scale.

F Supplementary Results and Scope

F.1 ID Accuracy at the Main Budget

Table 7 gives the numerical results underlying Figure 2.

Table 7: Mean ID answer accuracy (%) at equivalent rank 16, or 25% factor storage. Base and Full LoRA are references. Bold denotes the best selection method within each model.

Method	Qwen2.5-1.5B	Qwen3-4B	Gemma-3-4B	Llama-3-8B
Base	11.8	12.5	16.7	19.2
Full LoRA	67.0	73.3	70.8	79.2
SVD	42.3	64.6	63.3	57.5
Behavior	52.6	67.5	67.8	69.6
SAE-only	57.8	77.4	**81.3**	74.8
Joint	**60.8**	**79.6**	76.7	**82.4**
Random	14.2	13.3	10.1	17.1

F.2 Feature Interventions

Under fixed-token continuation, selected-feature insertion in Base improves answer accuracy by 5.38 points over no intervention and by 8.79 points over random-feature insertion. Selected-feature deletion in Full LoRA reduces state and answer accuracy by 14.97 and 22.38 points under free continuation, and by 12.38 and 24.72 points under fixed-token continuation. In the latter condition, answer accuracy is 12.63 points below random-feature deletion (Table 1). Carry-restoration contrasts are collected separately in Appendix D.4.

F.3 Limitations and Scope of Interpretation

The component scores measure local sensitivity along a uniform interpolation path. Interactions can change finite-removal effects. Comparisons with SVD also involve different allocation rules, as detailed in Appendix C.1.

Decoded SAE interventions preserve the host reconstruction residual and can affect multiple re-encoded coordinates. Fixed-token continuation supplies the correct immediate token before generation resumes.

The synthetic tasks cover column arithmetic and variable isolation. GSM8K provides a complementary evaluation of mathematical word problems. Joint exceeds Full LoRA on GSM8K for three pairs, yet remains below Base for all four (Table 2).