SPARSE WEIGHT DECOMPOSITION FOR EFFICIENT CIRCUIT EXTRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Dense pretrained transformers do not naturally expose interpretable units for circuit extraction. Existing approaches obtain such units by learning auxiliary sparse representations or training sparse models, incurring substantial additional computation while potentially introducing a fidelity gap between the representation being analyzed and the original pretrained model. We propose Sparse Weight Decomposition (SWD), which reparameterizes pretrained linear projections by factorizing each weight matrix into two sparse factors whose shared intermediate coordinates serve as individually addressable circuit units. Without training a separate replacement network, this parametric representation supports the same scoring, selection, and ablation circuit extraction workflow used for methods that learn sparse features. Across single-matrix replacements, SWD matches the held-out fidelity achieved by Transcoder and other strong baselines while using less than 1% of the data that those baselines use to train their replacements. At matched replacement fidelity, SWD reaches the same circuit sufficiency and necessity targets with fewer active read/write edges and selected units across tasks on GPT-2, Qwen2.5, and Qwen3.5-27B. We further show that SWD remains effective for full-model replacement of all attention and MLP weight matrices after fine-tuning the nonzero factor values. Finally, SWD also features a zero-data variant, allowing broader use of mechanistic interpretability analysis (*e.g.*, per-step analysis).

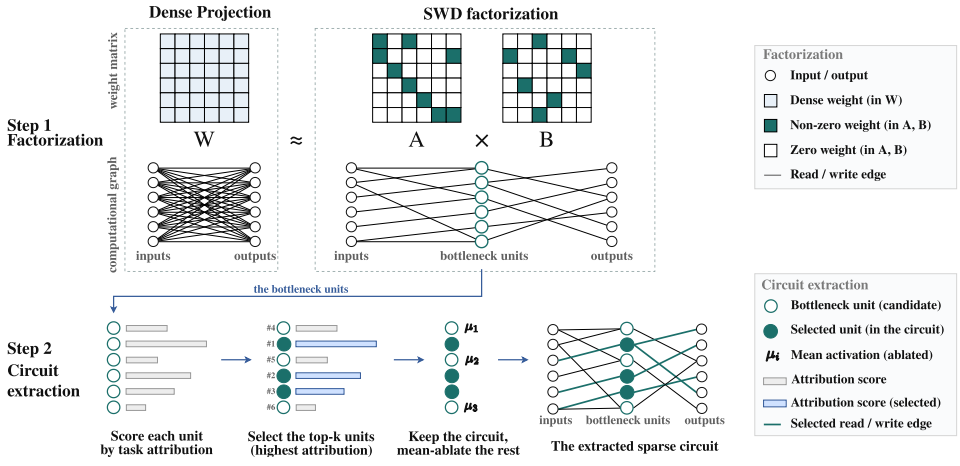

Figure 1: The SWD pipeline. **Step 1 (factorization).** A dense linear projection with weight matrix $\boldsymbol{W}$ is reparameterized by two sparse factors, $\boldsymbol{W} \approx \boldsymbol{AB}$. Each shared intermediate coordinate i is a *bottleneck unit*: it reads from the input through $\boldsymbol{A}_{:,i}$ and writes to the output through $\boldsymbol{B}_{i,:}$, defining the fixed rank-one path $\boldsymbol{A}_{:,i}\boldsymbol{B}_{i,:}$. In the graph representation, each nonzero factor entry is a directed read or write edge and a zero entry is no edge. For illustration, the diagram uses six bottleneck units. **Step 2 (circuit extraction).** These six units are scored and ranked by task attribution (bar length is the attribution score; labels #1–#6 give the ranking). The top-k units are selected for the circuit; here $k = 3$, and the activation of each unselected unit i is replaced by its mean activation μ_i.

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.

1 INTRODUCTION

Circuit analysis seeks a small set of internal units that is sufficient to reproduce a behavior and necessary in the sense that ablating it predictably degrades that behavior (Conmy et al., 2023; Syed et al., 2023; Bhaskar et al., 2024). Pretrained transformers, however, do not expose such units: they implement task behavior through dense linear projections, where individual neurons can be polysemantic and task-relevant effects are distributed across many activations and parameters (Elhage et al., 2022; Cunningham et al., 2023). This leaves a gap between the sparse causal explanations we seek and the dense computational substrate we aim to analyze.

Existing approaches obtain interpretable units for circuit analysis in several ways. Sparse autoencoders and Transcoders learn feature dictionaries or replacement modules in activation space; parameter-decomposition methods learn components of the model's weights; and sparse pretraining builds sparse connectivity into the model during pretraining (Cunningham et al., 2023; Dunefsky et al., 2024; Braun et al., 2025; Bushnaq et al., 2025; 2026; Gao et al., 2025). Each of these three families provides useful units for analysis, but requires training or optimizing an additional representation or model beyond the dense pretrained checkpoint. The reported overheads can be substantial. For example, weight-sparse models require 100–1000$\times$ more training and inference compute than dense models of comparable capability (Gao et al., 2025).

Alternatively, a natural question is whether circuit units can be obtained directly from a pretrained checkpoint, avoiding the cost of learning auxiliary representations altogether. NaNA (Xue & Andrzejak, 2026) applies exact SVD directly to MLP weights and shows that a small set of top-ranked components can recover the target prediction without training an auxiliary representation; however, each retained component still has dense input and output vectors, so retaining only a few components does not necessarily yield a compact circuit in terms of active connections. Weight-sparse transformers, trained from scratch, further show that sparse connectivity can yield substantially smaller and more human-understandable task circuits at comparable pretraining loss (Gao et al., 2025), suggesting that sparse connectivity is a useful structural constraint for circuit analysis. We therefore ask whether weight-side sparse connectivity can be introduced after pretraining by reparameterizing an existing dense checkpoint. To pursue this question, we draw on Double Sparse Factorization from model compression, which approximates a dense matrix as the product of two sparse matrices (Boza & Macko, 2025), and evaluate whether it can preserve the original model's behavior while yielding task circuits with fewer active connections.

Our key idea is to obtain circuit units directly from pretrained weights via *sparse factorization*, avoiding the cost of training an auxiliary representation. Specifically, we introduce Sparse Weight Decomposition (SWD), which reparameterizes a pretrained dense linear projection with weight matrix $\boldsymbol{W}$ as $\boldsymbol{W} \approx \boldsymbol{AB}$, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are sparse factor matrices. Each intermediate coordinate i is a *bottleneck unit*: it reads through $\boldsymbol{A}_{:,i}$ and writes through $\boldsymbol{B}_{i,:}$, defining a fixed rank-one path $\boldsymbol{A}_{:,i}\boldsymbol{B}_{i,:}$ with sparse read and write connections. Its scalar activation can be independently scored, selected, and ablated using the same circuit-extraction workflow applied to learned sparse features (Conmy et al., 2023; Bhaskar et al., 2024; Miller et al., 2024; Syed et al., 2023; Wang et al., 2022).

In our experiments, we evaluate both replacement fidelity (how closely each replacement preserves dense-model behavior before circuit extraction) and the circuit cost-quality tradeoff (how many selected units or active read/write edges are required to reach each sufficiency or necessity target). We first replace one MLP matrix in GPT-2 Small (Radford et al., 2019), three Qwen2.5 (Qwen et al., 2025) models, and Qwen3.5-27B (Qwen Team, 2026), comparing SWD with activation- and parameter-space baselines under matched pre-pruning cross-entropy (CE) loss (Section 3.3). We then replace all 48 attention and MLP weight matrices across the 12 transformer blocks of GPT-2 Small to test whether SWD works when local approximation errors accumulate (Section 3.4); use exact full-rank dense factorizations as controls that test whether low circuit cost comes from factorization alone or from sparse read/write connectivity (Section 3.5); and evaluate a zero-data variant that constructs factors from weights alone (Section 3.6).

Our contributions are as follows:

- We repurpose and evaluate sparse weight factorization as a circuit-extraction method: its bottleneck units can be scored and pruned without training a separate replacement network.

- Compared with baselines such as Transcoder and VPD, SWD reaches matched replacement fidelity using less than 1% as much data and, at that fidelity, SWD requires fewer active read/write edges to attain the same sufficiency and necessity thresholds.
- SWD scales across model sizes and, after fixed-support fine-tuning, remains effective when all attention and MLP weight matrices are replaced simultaneously. It also admits a variant that requires no calibration data.

2 METHODOLOGY

2.1 SPARSE WEIGHT DECOMPOSITION

Sparsity on the edges. A dense matrix has many factorizations, but an exact dense reparameterization such as Singular Value Decomposition (SVD) still leaves every bottleneck unit connected to almost every input and output dimension. Motivated by evidence that weight-sparse models preserve more tractable circuits (Gao et al., 2025), SWD places sparsity on the factorization's read and write *edges* (weights). Each bottleneck unit therefore reads from and writes to only a small set of input/output dimensions. Section 3.5 shows that the resulting cost-quality tradeoff advantage does not come from exact factorization alone.

Objective. For a dense matrix $\boldsymbol{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, SWD seeks sparse factors $\boldsymbol{A} \in \mathbb{R}^{d_{\text{in}} \times m}$ and $\boldsymbol{B} \in \mathbb{R}^{m \times d_{\text{out}}}$ such that $\boldsymbol{W} \approx \widehat{\boldsymbol{W}} = \boldsymbol{AB}$, where m is the number of bottleneck units. Table 1 in Appendix A summarizes the recurring notation used throughout the paper. Under a fixed nonzero budget K, the sparse product cannot in general preserve the action of $\boldsymbol{W}$ equally well in every input direction, so the optimization must decide where to allocate approximation error (Boza & Macko, 2025). The Frobenius objective $\|\boldsymbol{W} - \boldsymbol{AB}\|_F^2$ nevertheless weights all directions equally so that at the same sparsity, it can spend limited fitting capacity on directions rarely encountered by the model, and leave avoidably large error on the activations the model actually processes (Section 3.6). Our default objective therefore minimizes reconstruction error on calibration activations:

$$\min_{\boldsymbol{A},\boldsymbol{B}} \mathbb{E}_{\boldsymbol{h}\sim\mathcal{D}_{\text{cal}}} \|\boldsymbol{hW} - \boldsymbol{hAB}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{A}\|_0 + \|\boldsymbol{B}\|_0 \le K,$$

where $\boldsymbol{h}$ is a calibration-activation vector and K is the total number of nonzero factor entries. In experiments, we report K as a sparsity level $s = 1 - \frac{K}{\|\boldsymbol{W}\|_0} = 1 - \frac{K}{d_{in} \cdot d_{out}}$ (larger s, smaller K). Since

$$\begin{aligned}\mathbb{E}_{\boldsymbol{h}\sim\mathcal{D}_{\text{cal}}} \|\boldsymbol{hW} - \boldsymbol{hAB}\|_2^2 &= \mathbb{E}_{\boldsymbol{h}\sim\mathcal{D}_{\text{cal}}} (\boldsymbol{W} - \boldsymbol{AB})^\top (\boldsymbol{h}^\top \boldsymbol{h}) (\boldsymbol{W} - \boldsymbol{AB}) \\ &= (\boldsymbol{W} - \boldsymbol{AB})^\top \mathbb{E}_{\boldsymbol{h}\sim\mathcal{D}_{\text{cal}}} [\boldsymbol{h}^\top \boldsymbol{h}] (\boldsymbol{W} - \boldsymbol{AB}),\end{aligned}$$

the Gram matrix $\boldsymbol{G} = \mathbb{E}_{\boldsymbol{h}\sim\mathcal{D}_{\text{cal}}}[\boldsymbol{h}^\top \boldsymbol{h}]$ directs the limited fitting capacity toward input directions with greater mass under the model's activation distribution. When calibration activations are unavailable, we also have a zero-data variant: set $\boldsymbol{G} = \boldsymbol{I}$, reducing the objective to $\|\boldsymbol{W} - \boldsymbol{AB}\|_F^2$, which we evaluate in Section 3.6.

Solver. Because the ℓ_0-constrained objective is nonconvex and NP-Hard, we approximately optimize it using the Double Sparse Factorization (DSF) heuristic (Boza & Macko, 2025). We split the total budget into fixed factor-specific budgets, $K_A + K_B = K$, and alternate between updating $\boldsymbol{A}$ with $\boldsymbol{B}$ fixed and updating $\boldsymbol{B}$ with $\boldsymbol{A}$ fixed. Each block update is an ℓ_0-constrained regression problem solved with the Alternating Direction Method of Multipliers (ADMM), with hard thresholding enforcing the prescribed nonzero budget (Boyd et al., 2011). After the alternating updates, DSF freezes the per-matrix supports and $\boldsymbol{A}$, and locally refits the surviving entries of $\boldsymbol{B}$ against the activation-weighted reconstruction objective. For a single matrix, this is the entire SWD process. When many weight matrices are replaced simultaneously, local approximation errors can accumulate. We therefore apply fixed-support fine-tuning: we minimize next-token cross-entropy by updating only the nonzero factor values, while keeping all other pretrained parameters and the zero/nonzero patterns of $\boldsymbol{A}$ and $\boldsymbol{B}$ fixed. This preserves the sparse read/write connectivity and its edge count. We refer to the resulting replacement as SWD-FT and evaluate it in Section 3.4; detailed solver and fine-tuning settings are given in Appendix D.2.

2.2 Circuit Extraction from Sparse Factors

The factorized replacement can be written as a sum of additive read–write paths. For an input-activation vector $\boldsymbol{h}$, define

$$z_i(\boldsymbol{h}) = \boldsymbol{h}\boldsymbol{A}_{:,i}, \qquad \boldsymbol{h}\widehat{\boldsymbol{W}} = \boldsymbol{h}\boldsymbol{A}\boldsymbol{B} = \sum_{i=0}^{m-1} z_i(\boldsymbol{h})\boldsymbol{B}_{i,:}.$$

As shown in Figure 1, bottleneck unit i reads $\boldsymbol{h}$ through the sparse vector $\boldsymbol{A}_{:,i}$ and writes the resulting scalar along the sparse vector $\boldsymbol{B}_{i,:}$; their supports therefore specify which input and output coordinates the path touches. Moreover, replacing $z_i(\boldsymbol{h})$ changes only the ith term in the sum, leaving the other paths unchanged. This makes each bottleneck unit individually addressable for circuit interventions.

We rank bottleneck units by a task-margin attribution score and evaluate nested top-k prefixes (Appendix D.3). Given this ranking, a candidate circuit is a subset $S \subseteq \{0, \ldots, m-1\}$. Let μ_i be the reference activation for bottleneck unit i. Following the mean-ablation convention used in prior circuit analysis (Wang et al., 2022), we set μ_i to be the mean activation on the circuit extraction training split. We form two complementary interventions:

$$z_i^{\text{keep}}(\boldsymbol{h}; S) = \begin{cases} z_i(\boldsymbol{h}), & i \in S, \\ \mu_i, & i \notin S, \end{cases} \qquad z_i^{\text{abl}}(\boldsymbol{h}; S) = \begin{cases} \mu_i, & i \in S, \\ z_i(\boldsymbol{h}), & i \notin S. \end{cases}$$

The keep intervention tests sufficiency: how much task behavior is retained by only keeping those units in S. The ablation intervention tests necessity: how much task behavior is lost by removing those units in S. In both cases, the intervened output is obtained by substituting the corresponding activations into $\sum_i z_i \boldsymbol{B}_{i,:}$.

These interventions are comparable across methods only when the unpruned replacements share the same pre-pruning behavior; otherwise scores are measured against different reference computations. The detailed replacement-fidelity matching rule and the common attribution-based S selection protocol are specified in Section 3.2.

3 Experiments

Our experiments track two criteria throughout: replacement fidelity and circuit cost-quality tradeoff, which is defined in Section 3.2. We first replace one MLP matrix in each of GPT-2 Small and the Qwen2.5 models from 0.5B to 3B with SWD, Transcoder, and VPD variants, then extend this single-matrix comparison to Qwen3.5-27B to test scalability, and evaluate sufficiency and necessity drop under matched pre-pruning cross-entropy. We then report full-model replacement results, isolate the contribution of sparse read/write edges from exact dense controls, evaluate a zero-data variant, and qualitatively inspect selected units and circuit.

3.1 Baselines

We compare SWD with three approaches that provide units for circuit analysis: **Transcoders** learn sparse activation-space replacements for MLP computations (Dunefsky et al., 2024); **sparse-pretrained models** impose weight sparsity during pretraining (Gao et al., 2025); and **VPD** learns parameter components with an input-dependent selector (Bushnaq et al., 2026). We evaluate two VPD configurations. **VPD-KL** denotes the original VPD configuration, whose fidelity term is model-level Kullback–Leibler (KL) divergence. **VPD-Recon-CI** retains the same component parameterization and causal-importance (CI) selector but replaces this term with local activation reconstruction. VPD-KL does not reach the fidelity required for CE-matched circuit comparison in our single-matrix replacement experiments, so we report it only for replacement quality; VPD-Recon-CI is used for circuit comparisons (Appendix D.5.2 and E.1). Since sparse pretraining produces an entire sparse model rather than a single-matrix replacement, we compare against it only in the full-model replacement experiment (Section 3.4). Table 5 summarizes how these methods differ in representation, sparsity, optimization, data requirements, and applicable scope. Full configurations are collected in Appendix D.5.

3.2 Evaluation Protocol

Our evaluation protocol largely follows the task-based circuit-discovery setup used by ACDC/auto-circuit and Edge-Pruning (Conmy et al., 2023; Bhaskar et al., 2024; Miller et al., 2024), together with first-order attribution scoring from attribution patching (Syed et al., 2023). Evaluation proceeds in two stages. First, we evaluate replacement fidelity using CE delta (replacement CE minus dense-model CE), supplemented where applicable by KL and activation relative mean-squared error (relative MSE; the relative error between the dense and replacement outputs of the replaced projection). Detailed definitions of the companion metrics are provided in Appendix E.1. In the replacement and circuit comparisons, all methods use FineWeb-Edu for training or calibration, with a disjoint split reserved for evaluation (Lozhkov et al., 2024). We report token–CE loss curves to evaluate data efficiency. Since circuit scores are computed relative to each method's own replacement, unless explicitly stated otherwise, we compare circuit performance only between replacements whose pre-pruning CE differs by at most 0.001. Second, for circuit extraction, we use greater-than, indirect-object identification (IOI), docstring, and gendered-pronoun task families from public Edge-Pruning and auto-circuit datasets (Bhaskar et al., 2024; Conmy et al., 2023; Miller et al., 2024). For each task τ, we compute unit scores and mean-ablation statistics on the circuit extraction training split (`circuit_train`). The resulting ranking defines a nested family of top-k prefixes S_k. We then evaluate these fixed prefixes on the held-out test split (`circuit_test`) and report held-out sufficiency and necessity drop as functions of circuit cost. Data sources, split sizes, and the Qwen task subsets are detailed in Appendix D.4.

For each method, we apply this common selection protocol to the candidate units exposed by its representation. Units are ranked by positive mean-centered first-order task-margin attribution (Appendix D.3). Ablated units are replaced by their mean activation on `circuit_train`; thus $M_{\text{keep}}(S)$ retains the selected units and mean-ablates the remainder, whereas $M_{\text{abl}}(S)$ mean-ablates the selected units and retains the remainder. Following prior causal circuit analyses (Wang et al., 2022), we use mean ablation for the main comparisons because it removes input-dependent variation while preserving each unit's average activation level. Because circuit evaluations can be sensitive to the ablation methodology, including the value assigned to ablated activations (Miller et al., 2024), we also test whether our conclusions depend on this choice. Appendix E.3 evaluates the GPT-2 single-matrix circuit under the zero-reference ranking and zero ablation. Let $g_\tau(M, \boldsymbol{x})$ be the task logit margin for example $\boldsymbol{x}$: the model's average logit for correct task answers minus its average logit for distractor answers, and let $Q_\tau(M; \mathcal{D}) = \mathbb{E}_{\boldsymbol{x}\in\mathcal{D}}[g_\tau(M, \boldsymbol{x})]$ be the average task score of model M on split $\mathcal{D}$, with M_{unpruned} denoting the replacement before circuit pruning. We report method-relative sufficiency $\text{Suff}_\tau(S; \mathcal{D}) = Q_\tau(M_{\text{keep}}(S); \mathcal{D})/Q_\tau(M_{\text{unpruned}}; \mathcal{D})$ and necessity drop $\text{NecDrop}_\tau(S; \mathcal{D}) = Q_\tau(M_{\text{unpruned}}; \mathcal{D}) - Q_\tau(M_{\text{abl}}(S); \mathcal{D})$. We measure circuit size by selected units $C_{\text{unit}}(S) = |S|$ and active edges $C_{\text{edge}}(S)$, the total number of nonzero read/write weights over the valid selected units; the definitions are given in Appendix D.3. An active edge is a nonzero weight in the read or write vector of a selected unit, and the active-edge count is the same whether S is retained for sufficiency or ablated for necessity. For each sufficiency or necessity value, we report the minimum selected units or minimum active edges among the evaluated top-k sets that reach that value.

3.3 Single-Matrix Replacement

We begin by replacing the GPT-2 Small layer-8 MLP output projection, `mlp.c_proj`, with each method's corresponding sparse representation. Figure 2 plots CE delta from the dense model against data used. SWD reaches low replacement error with substantially less data: SWD (s=0.5) reaches low CE delta after a few thousand tokens versus roughly 10^6 for the baselines, whereas VPD-KL improves only after much longer optimizer replay and then plateaus. Because its CE cannot be matched, we retain VPD-KL as a replacement-quality reference but exclude it from circuit extraction. Appendix E.1 documents activation relative MSE, KL, and full settings.

Figure 3 compares performance at matched pre-pruning CE, with one sparser SWD setting shown separately as a non-matched control. Across all four tasks, SWD attains the same sufficiency and necessity targets with substantially fewer active edges than Transcoder and VPD-Recon-CI. This advantage is not specific to mean ablation: under zero ablation, SWD continues to require fewer active edges across all four tasks (Appendix E.3). Selected-unit circuit results and settings are in Appendix E.2 and D.5.

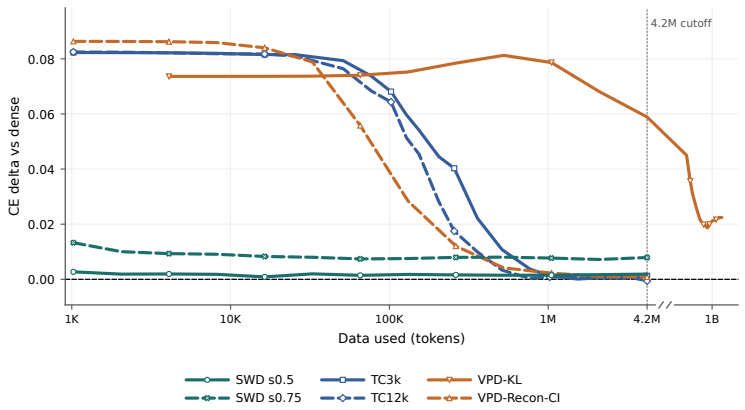

Figure 2: GPT-2 single-matrix replacement quality. SWD reaches low CE delta with far less data; VPD-KL is a replacement-quality reference, excluded from circuit extraction. See Appendix E.1 for companion metrics and settings.

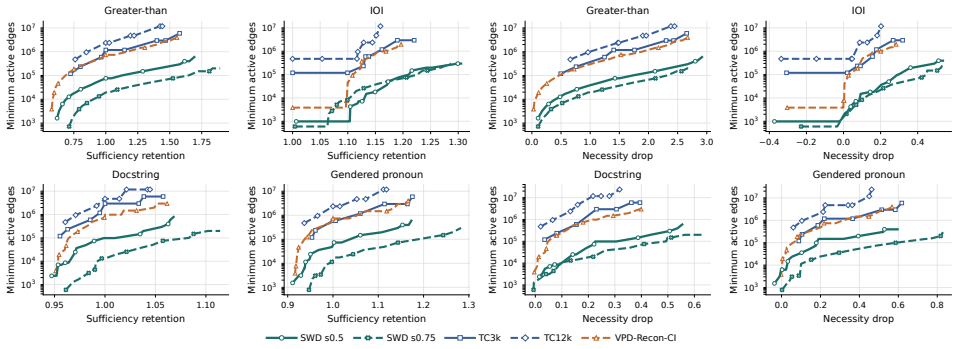

Figure 3: GPT-2 single-matrix circuit results. SWD reaches the target thresholds with markedly fewer active edges than the Transcoder and VPD-Recon-CI baselines.

The two SWD settings expose a trade-off between replacement fidelity and circuit size: s=0.5 achieves lower replacement CE and is used in the CE-matched comparison, whereas the more aggressive s=0.75 incurs a modest fidelity penalty but requires fewer active edges to reach the same task behavior. Thus, increasing factor sparsity can reduce circuit size even when it slightly reduces replacement fidelity.

We repeat the single-matrix comparison on Qwen2.5 models from 0.5B to 3B and extend it to Qwen3.5-27B. On both Qwen2.5-3B and Qwen3.5-27B, SWD reaches low CE delta with substantially fewer tokens than the trained baselines (Appendix E.5, Figure 21). Figure 4 shows the same circuit-cost trend observed on GPT-2: SWD reaches comparable sufficiency and necessity with fewer active edges, extending this advantage to Qwen3.5-27B. The Qwen2.5 circuit results (0.5B to 3B) are reported in Appendix E.4, with further details for Qwen3.5-27B in Appendix E.5.

In the experiments above, each method replaces only one MLP matrix. Appendix G additionally reports an MLP-replacement experiment, where the standard Transcoder replaces the entire MLP and SWD replaces both MLP weight matrices with sparse factorizations while retaining the original GELU.

3.4 Full-Model Replacement

In this experiment, we replace all 48 attention and MLP weight matrices across the 12 transformer blocks of GPT-2 Small. The embeddings, layer-normalization modules, nonlinearities, and LM head remain unchanged. We compare with weight-sparse pretraining (Gao et al., 2025) at an approximately matched budget of 26.77M active weights (about 68% sparsity relative to the

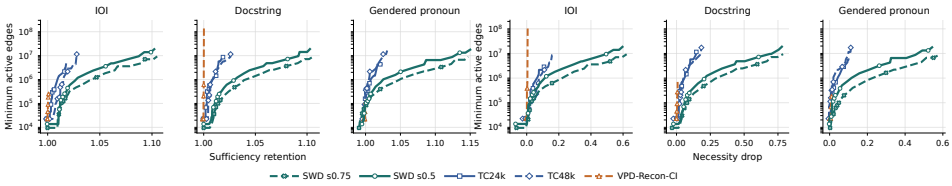

Figure 4: Qwen3.5-27B single-matrix circuit results. The vertical axis reports the minimum active edges. Checkpoint settings and the selected-unit companion are reported in Appendix E.5.

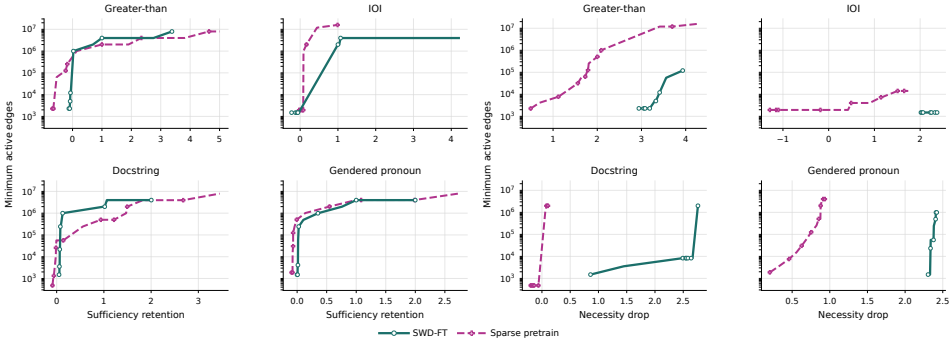

Figure 5: GPT-2 Small full-model replacement circuit results. The two methods are broadly comparable in sufficiency, while SWD-FT reaches larger necessity drops with fewer active edges on all four tasks.

dense transformer-body weight matrices); the exact training recipe and accounting are given in Appendix D.5.3. Approximation error now accumulates across layers: the sparse factorization reaches held-out CE 3.90. We therefore apply fixed-support fine-tuning, updating only the nonzero factor values while keeping the sparse supports (and hence the edge count) fixed. The resulting SWD-FT replacement reaches CE 3.44, matching the sparse-pretraining checkpoint's 3.45. Because SWD-FT starts from the pretrained dense checkpoint, it needs only 20.6M tokens for factorization and fine-tuning, whereas sparse pretraining trains from scratch on 2.884B tokens. Appendix F gives the full curve.

For task circuits, we prune SWD bottleneck units and the sparse-pretrained model's latent channels under their corresponding protocols (Section 3.2) and compare the minimum active edges needed to reach each held-out target (Figure 5). At the approximately matched budget, the two methods are broadly comparable in sufficiency, whereas SWD-FT reaches larger necessity drops with fewer active edges on all four tasks.

3.5 Ablation: Exact Dense Reparameterizations

The fact that SWD requires fewer circuit edges could arise either from its sparse read/write matrices or simply from factorizing the original matrix into intermediate units. To distinguish these explanations, we compare SWD with two exact factorizations of the same GPT-2 Small output-projection matrix. The first is a full-rank SVD, using the same exact rank-one decomposition as NaNA (Xue & Andrzejak, 2026); the second is Random-B, which uses a seeded random orthogonal matrix for $\boldsymbol{B}$ and computes $\boldsymbol{A}$ from $\boldsymbol{W}$ and $\boldsymbol{B}$. Both factorizations reproduce the target matrix exactly, but their factor matrices are dense. We apply the same task-level attribution and mean-ablation procedure to all three methods. Figure 6 shows that the exact SVD and Random-B factorizations require more active edges than SWD to retain the same task behavior under sufficiency evaluation or induce the same performance drop under necessity evaluation. Since this pattern holds for both the SVD basis and random orthogonal bases, the comparison isolates the contribution of sparse read/write structure from factorization alone. Appendix H provides the construction, fidelity checks, and companion selected-unit results.

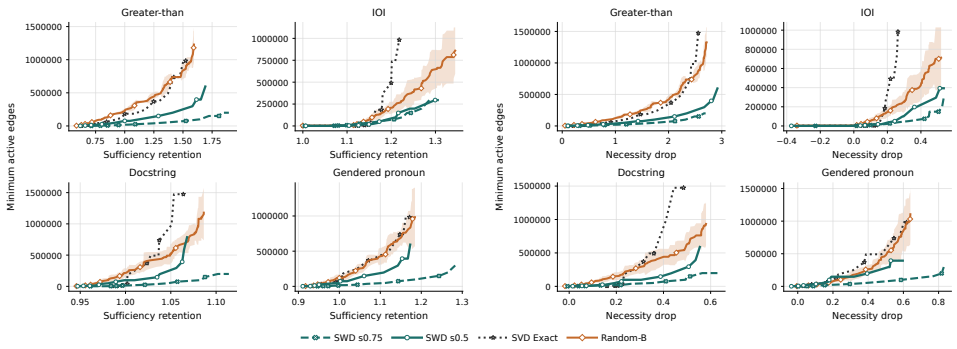

Figure 6: Exact dense reparameterization controls for the GPT-2 output-projection surface. The comparison with SWD tests whether sparse read/write structure reduces the number of active edges beyond factorization alone. See Appendix H for the control settings.

3.6 Zero-Data Factorization

The previous section shows that sparse structure is what enables SWD to yield compact circuits. We next ask whether useful bottleneck units can be obtained without calibration activations. Calibration activations may be unavailable and can tie the factorization to a particular data distribution, so we evaluate the zero-data SWD variant defined in Section 2.1, which depends only on checkpoint weights and can in principle be applied at every training step. On the GPT-2 output-projection surface, we vary s from 0.125 to 0.875 and measure two complementary notions of fidelity: relative Frobenius error $\|\boldsymbol{W} - \boldsymbol{AB}\|_F / \|\boldsymbol{W}\|_F$ in parameter space and held-out CE delta in model behavior.

As shown in Figure 28 (Appendix I), zero-data SWD remains closer to $\boldsymbol{W}$ throughout all sparsity levels, reaching relative error 0.366 versus 0.415 for activation-calibrated SWD at $s = 0.75$. Conversely, activation-calibrated SWD achieves lower CE under aggressive sparsification because it allocates the limited edge budget toward the distribution of activation directions. Thus zero-data SWD better preserves the raw weights, whereas calibration better preserves behavior on typical model activations. To further analyze the circuit properties of the zero-data factors, we apply the same circuit extraction protocol in Appendix I.2 and show that zero-data bottlenecks remain useful for task circuits.

3.7 Qualitative Results

We conclude the experimental analysis with a qualitative view of SWD bottleneck units, examining whether units selected by task attribution have recognizable semantic hypotheses. Appendices K, L, and M further use SWD factors for a targeted directional edit, a query–key analysis of an attention head, and interactions among MLP bottleneck units. These analyses complement the aggregate circuit results above by illustrating what the extracted units represent and how they can be manipulated.

We audit GreaterThan on the GPT-2 Small all-layer `mlp.c_proj` surface. We rank all $9{,}208$ bottleneck units by the train-split positive first-order attribution (Appendix D.3), and Figure 7 shows the two highest-ranked units in layers 6, 8, and 10; all six lie in the global top-512 prefix. For each unit, we inspect its 20 highest-magnitude activation contexts from a task-independent $2{,}048$-token WikiText-2 dashboard and assign a semantic hypothesis from recurring patterns; Appendix J gives the full protocol and per-layer ranks. The resulting task-independent labels align with the task family: four of the six highlighted paths are associated with numbers, quantities, or measurements, while the remaining two capture syntax or named entities. This correspondence gives the extracted circuit a semantic interpretation that complements the held-out sufficiency and necessity results above.

4 Related Work

Mechanistic interpretability (MI) seeks small computational subgraphs whose interventions explain a model behavior, and ACDC, attribution patching, and edge pruning make the search over com-

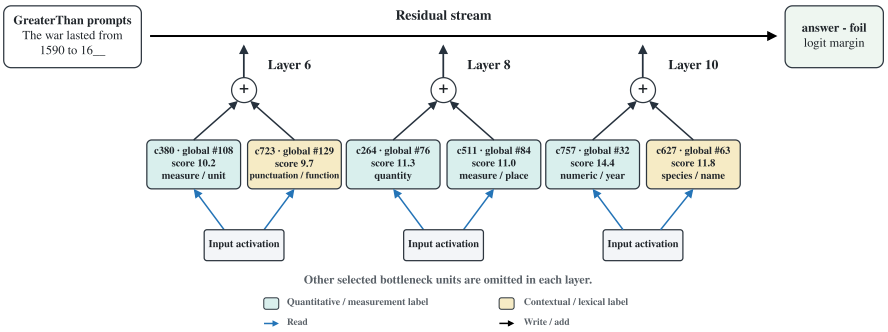

Figure 7: Semantic audit of high-attribution paths in a GreaterThan SWD circuit. Nodes are selected by task-related first-order attribution; their semantic labels come from recurring patterns in task-independent, high-activation contexts and are not used for selection. Four of the six displayed units have number-, quantity-, or measurement-related hypotheses.

ponents or edges more scalable (Olah et al., 2020; Conmy et al., 2023; Syed et al., 2023; Bhaskar et al., 2024). Whereas these methods primarily decide which nodes or edges to retain in an existing computational basis, SWD introduces an alternative basis of addressable bottleneck units inside otherwise dense linear maps. Sparse autoencoders and Transcoders instead learn a new activation dictionary or replacement module on an activation corpus (Cunningham et al., 2023; Dunefsky et al., 2024; Ameisen et al., 2025), whereas SWD factorizes a checkpoint matrix and sparsifies each unit's parameter-side read and write vectors, defining each unit as a sparse rank-one path without assuming that it is a monosemantic activation feature. Parameter-decomposition methods (APD, SPD, VPD) and NaNA also seek additive or rank-one components in model weights (Braun et al., 2025; Bushnaq et al., 2025; 2026; Xue & Andrzejak, 2026), but SWD's components consist of one sparse column of $\boldsymbol{A}$ and one sparse row of $\boldsymbol{B}$, constructed independently of the downstream circuit task. Finally, weight pruning and sparse pretraining show that sparse structure can be recovered from or built into models (Frantar & Alistarh, 2023; Sun et al., 2024; Gao et al., 2025); SWD instead introduces bottleneck units with sparse edges post hoc by adopting Double Sparse Factorization (Boza & Macko, 2025). An extended discussion is given in Appendix B.

5 Limitations

SWD makes local circuit hypotheses substantially cheaper to extract and test, but it does not eliminate the broader limitations of mechanistic interpretability. Efficient extraction is not complete understanding: our sufficiency and necessity results establish that selected bottleneck units causally affect held-out task margins, but do not explain the model globally (Hendrycks & Hiscott, 2025). The extracted circuits are also local and non-unique, depending on the objective, calibration distribution, and sparsity budget, and are computed on an approximate replacement $\boldsymbol{AB}$ under a first-order ranking and mean-ablation protocol whose conclusions can vary with the extraction choices (Zheng et al., 2021; Miller et al., 2024; Sheng & Fu, 2026). Finally, scaling verification remains open. Appendix C discusses these limitations in full.

6 Conclusion

Sparse Weight Decomposition turns dense pretrained projections into bottleneck units with sparse read/write connectivity that can be used directly for circuit extraction. At matched replacement fidelity, SWD achieves comparable circuit sufficiency and necessity with fewer selected units and active edges than learned sparse replacements across GPT-2, Qwen2.5, and Qwen3.5-27B. Fixed-support fine-tuning extends it to full-model replacement and a zero-data variant recovers useful units from weights alone, giving a parameter-side surface for compact circuit selection in dense models.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code development, qualitative analysis of activation contexts, and manuscript polishing. All AI-assisted outputs were reviewed by the authors. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Implementation details, data sources and splits, baseline configurations, factorization objectives, unit-scoring procedures, and evaluation protocols are provided in the appendices. The reported results are accompanied by the model settings, checkpoint specifications, and experiment configurations used for the comparisons. Anonymized code, experiment configurations, and plot-level result tables are provided in the supplementary material.

REFERENCES

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, et al. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011. doi: 10.1561/2200000016.

Vladimir Boza and Vladimir Macko. Two sparse matrices are better than one: Sparsifying neural networks with double sparse factorization. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=DwiwOcK1B7.

Dan Braun, Lucius Bushnaq, Stefan Heimersheim, Jake Mendel, and Lee Sharkey. Interpretability in parameter space: Minimizing mechanistic description length with attribution-based parameter decomposition. *arXiv preprint arXiv:2501.14926*, 2025.

Trenton Bricken, Adly Templeton, Joshua Batson, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features.

Lucius Bushnaq, Dan Braun, and Lee Sharkey. Stochastic parameter decomposition. *arXiv preprint arXiv:2506.20790*, 2025.

Lucius Bushnaq, Dan Braun, Oliver Clive-Griffin, Bart Bussmann, Nathan Hu, Michael Ivanitskiy, Linda Linsefors, and Lee Sharkey. Interpreting language model parameters, April 2026. URL https://www.goodfire.ai/research/interpreting-lm-parameters.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adria Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *arXiv preprint arXiv:2304.14997*, 2023.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Alexandre d'Aspremont, Laurent Ghaoui, Michael Jordan, and Gert Lanckriet. A direct formulation for sparse pca using semidefinite programming. *Advances in neural information processing systems*, 17, 2004.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. *arXiv preprint arXiv:2406.11944*, 2024.

Nelson Elhage, Neel Nanda, Catherine Olsson, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Nelson Elhage, Tristan Hume, Catherine Olsson, et al. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Gabriel Franco and Mark Crovella. Sparse attention decomposition applied to circuit tracing, 2024. URL https://arxiv.org/abs/2410.00340.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 2023. URL https://proceedings.mlr.press/v202/frantar23a.html.

Leo Gao, Achyuta Rajaram, Jacob Coxon, Soham V. Govande, Bowen Baker, and Dan Mossing. Weight-sparse transformers have interpretable circuits, 2025. URL https://arxiv.org/abs/2511.13653.

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

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=TZ0CCGDcuT.

Dan Hendrycks and Laura Hiscott. The misguided quest for mechanistic AI interpretability. *AI Frontiers*, May 2025. URL https://ai-frontiers.org/articles/the-misguided-quest-for-mechanistic-ai-interpretability. Guest Commentary.

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. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Harish Kamath, Emmanuel Ameisen, Isaac Kauvar, Rodrigo Luger, Wes Gurnee, Adam Pearce, Sam Zimmerman, Joshua Batson, Thomas Conerly, Chris Olah, and Jack Lindsey. Tracing attention computation through feature interactions. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attention-qk/index.html.

Andrey Kuzmin, Mart van Baalen, Markus Nagel, and Arash Behboodi. Quantized sparse weight decomposition for neural network compression. *arXiv preprint arXiv:2207.11048*, 2022.

Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. doi: 10.1038/44565.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, Janos Kramar, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma Scope: Open sparse autoencoders everywhere all at once on Gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 278–300. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.blackboxnlp-1.19/.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(1), 2010.

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 *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I4e82CIDxv.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. *arXiv preprint arXiv:2202.05262*, 2022.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Joseph Miller, Bilal Chughtai, and William Saunders. Transformer circuit faithfulness metrics are not robust. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=zSf8PJyQb2.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fiotto-Kaufman, Tal Haklay, Michael Hanna, Jing Huang, Rohan Gupta, Yaniv Nikankin, Hadas Orgad, Nikhil Prakash, Anja Reusch, Aruna Sankaranarayanan, Shun Shao, Alessandro Stolfo, Martin Tutek, Amir Zur, David Bau, and Yonatan Belinkov. MIB: A mechanistic interpretability benchmark. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45069–45108. PMLR, 2025. URL https://proceedings.mlr.press/v267/mueller25a.html.

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

Catherine Olsson, Nelson Elhage, Neel Nanda, et al. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Yang Sheng and Jie Fu. Circuit claims depend on what is extracted and how it is compared, 2026. URL https://arxiv.org/abs/2607.18921.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

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

Michael Tsang, Dehua Cheng, and Yan Liu. Detecting statistical interactions from neural network weights. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ByOfBggRZ.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=LNYIUouhdt.

Min Xue and Artur Andrzejak. SVD as a fast interpretability method for transformers. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=7tt8TwMjdJ.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023. URL https://arxiv.org/abs/2312.05821.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=9eJv5PS27Q.

Léon Zheng, Elisa Riccietti, and Rémi Gribonval. Identifiability in exact two-layer sparse matrix factorization. *arXiv preprint arXiv:2110.01235*, 2021. URL https://arxiv.org/abs/2110.01235.

A NOTATION

Table 1 consolidates the recurring notation used across the main text and appendices. We typeset scalars and indices in ordinary italic, vectors in bold lowercase, and matrices in bold uppercase; sets, datasets, models, and scalar-valued functions remain nonbold. Layer, head, and unit identifiers are zero-based, whereas displayed attribution ranks are one-based. For method-specific terms, we call an SWD intermediate coordinate a *bottleneck unit*, a Transcoder hidden element a *feature*, a VPD element a *parameter component*, and a sparse-pretrained hidden element a *latent channel*. We use *unit* as the method-agnostic umbrella term in cross-method comparisons and selected-unit figures.

Table 1: Notation used throughout the paper.

Symbol	Meaning
$\boldsymbol{W}, \widehat{\boldsymbol{W}}$	Dense target weight matrix and its SWD replacement, $\widehat{\boldsymbol{W}} = \boldsymbol{A}\boldsymbol{B}$.
$\boldsymbol{A}, \boldsymbol{B}$	Sparse read and write factors, $\boldsymbol{A} \in \mathbb{R}^{d_{\text{in}} \times m}$ and $\boldsymbol{B} \in \mathbb{R}^{m \times d_{\text{out}}}$.
$d_{\text{in}}, d_{\text{out}}$	Input and output dimensions of the target matrix.
m, C	Numbers of SWD bottleneck units and VPD component slots, respectively.
i, k	i indexes a candidate unit; k is the number of highest-ranked units retained in a top-k circuit.
$\boldsymbol{x}, \boldsymbol{h}_{\ell,t}(\boldsymbol{x})$	Input text example $\boldsymbol{x}$ and the input activation to target module ℓ at token position t.
$\boldsymbol{H}$	Calibration-activation matrix whose rows are input-activation vectors $\boldsymbol{h}$.
$\boldsymbol{G}, \boldsymbol{I}$	Calibration Gram matrix $\boldsymbol{G} = \mathbb{E}[\boldsymbol{h}^\top \boldsymbol{h}]$ and the identity matrix used by the zero-data objective.
K, s	Total factor-entry budget $K = \|\boldsymbol{A}\|_0 + \|\boldsymbol{B}\|_0$ and factor sparsity $s = 1 - K/\|\boldsymbol{W}\|_0$.
$\mathcal{D}_{\text{cal}}$	Distribution of calibration activations used to fit SWD.
$\mathcal{D}_{\tau,\text{train}}, \mathcal{D}_{\tau,\text{test}}$	Training and held-out evaluation splits for circuit task τ.
z_i, μ_i, a_i	Activation of unit i, its train-split mean used for ablation, and its task-attribution score. For SWD, $z_i = \boldsymbol{h}\boldsymbol{A}_{:,i}$.
S_k, S_{eff}	S_k is the set of the k highest-ranked units; S_{eff} contains the selected units with at least one nonzero read edge and one nonzero write edge.
$\mathbf{r}_i, \mathbf{w}_i$	Method-specific read and write weight vectors of unit i; for SWD these are $\boldsymbol{A}_{:,i}$ and $\boldsymbol{B}_{i,:}$.
$C_{\text{unit}}(S), C_{\text{edge}}(S)$	Selected-unit cost $\|S\|$ and active-edge cost $\sum_{i \in S_{\text{eff}}} (\|\mathbf{r}_i\|_0 + \|\mathbf{w}_i\|_0)$.
M_{unpruned}	Replacement model before circuit pruning.
$M_{\text{keep}}(S), M_{\text{abl}}(S)$	Models that respectively retain only S or mean-ablate S.
$g_\tau(M, \boldsymbol{x})$	Per-example answer-versus-distractor logit margin for task τ.
$Q_\tau(M; \mathcal{D})$	Average task margin of model M over split $\mathcal{D}$.

The support of a factor is the set of its nonzero entries, and its factor mask is the corresponding binary indicator. A unit is valid when both its read and write vectors contain at least one nonzero entry; S_{eff} is the valid subset of a selected set.

B EXTENDED RELATED WORK

Task-level circuit discovery and evaluation. Mechanistic interpretability (MI) seeks small computational subgraphs whose interventions explain a model behavior (Olah et al., 2020). Manual transformer-circuit analyses have isolated mechanisms such as induction heads, indirect-object identification, and greater-than comparison (Elhage et al., 2021; Olsson et al., 2022; Wang et al., 2022;

Hanna et al., 2023). Causal tracing and activation patching localize behavior by exchanging or ablating internal states, while ACDC, attribution patching, and edge pruning make the search over components or edges more scalable (Meng et al., 2022; Conmy et al., 2023; Syed et al., 2023; Bhaskar et al., 2024). Recent evaluations emphasize that circuit conclusions depend on the intervention, metric, and granularity, and therefore favor held-out causal tests and concise recovered pathways over overlap with a single presumed circuit (Hanna et al., 2024; Zhang & Nanda, 2024; Miller et al., 2024; Mueller et al., 2025). Whereas these methods primarily decide which nodes or edges to retain in an existing computational basis, SWD introduces an alternative basis of addressable bottleneck units inside otherwise dense linear maps. It is thus complementary to circuit-search algorithms: attribution and mean ablation can be applied to the induced units, with held-out sufficiency and necessity measuring circuit quality and selected units and active edges measuring circuit size.

Sparse activation features and learned replacement modules. Superposition makes native neurons polysemantic and motivates learning overcomplete sparse feature dictionaries (Elhage et al., 2022). Sparse autoencoders (SAEs) decompose residual-stream or sublayer activations into sparsely active features, and large releases such as Gemma Scope make such dictionaries available across layers and model scales (Bricken et al., 2023; Cunningham et al., 2023; Lieberum et al., 2024). Sparse feature circuits connect SAE features into causal graphs that support intervention and editing (Marks et al., 2025). Transcoders instead learn a sparse-feature module that approximates an MLP's input–output computation, avoiding some difficulties in tracing an SAE feature through the original nonlinear MLP; attribution graphs extend this replacement-based view to larger computation graphs (Dunefsky et al., 2024; Ameisen et al., 2025; Kamath et al., 2025). Like these approaches, SWD exposes scalar units that can be scored, selected, and intervened on. SAEs and Transcoders, however, fit a new activation dictionary or replacement network on an activation corpus, whereas SWD factorizes a checkpoint matrix and sparsifies each unit's parameter-side read and write vectors. This construction applies uniformly to MLP and attention projections, and defines each unit as a sparse rank-one path without assuming that it is a monosemantic activation feature. We test the practical effect of this distinction through the matched-fidelity Transcoder comparison.

Parameter-space interpretability. Parameter-decomposition methods seek simpler additive components directly in model weights. Attribution-based Parameter Decomposition (APD) optimizes for parameter faithfulness, sparse per-input component use, and component simplicity; Stochastic Parameter Decomposition (SPD) develops a more scalable stochastic formulation (Braun et al., 2025; Bushnaq et al., 2025). VPD extends this family to language-model parameters using an input-dependent selector, and sparse attention decomposition uses SVD-derived parameter components to trace query–key computations (Bushnaq et al., 2026; Franco & Crovella, 2024). NaNA treats each rank-one component in the exact SVD of an MLP weight matrix as a detector–effector unit (DEU). It scores each DEU by how strongly it is activated by the input and how much it supports a specified target token, then retains or removes the highest-scoring units to test their effect on the prediction (Xue & Andrzejak, 2026). SWD also constructs fixed rank-one components from model weights, but each component consists of one sparse column of $\boldsymbol{A}$ and one sparse row of $\boldsymbol{B}$. SWD constructs these components independently of the downstream circuit task and subsequently ranks them using a common task-margin attribution protocol; VPD instead learns input-dependent component use as part of the decomposition.

Sparse models and post-training pruning. Weight pruning shows that substantial parameter sparsity can be recovered from dense language models with little or no retraining. SparseGPT uses approximate second-order information for one-shot pruning, while Wanda scores weights using both magnitude and observed input activations (Frantar & Alistarh, 2023; Sun et al., 2024). A complementary approach builds weight and activation sparsity into pretraining; such sparse-pretrained transformers can preserve more tractable circuits than comparable dense models, albeit at the cost of training a new model (Gao et al., 2025). These results motivate the hypothesis that useful sparse structure can be recovered from a dense checkpoint, but do not directly provide the intervention surface used by SWD. Pruning removes scalar edges without creating intermediate units that can be independently scored and ablated, while sparse pretraining changes the model from the start. SWD instead introduces bottleneck units with sparse incoming and outgoing edges post hoc. The sparse-

pretraining comparison tests the shared sparse-connectivity hypothesis, while the exact dense-factor controls distinguish the effect of sparse edges from that of factorization alone.

Matrix factorization and compression. Classical matrix methods impose low-rank, non-negativity, or sparsity to obtain compact representations (Lee & Seung, 1999; Mairal et al., 2010; d'Aspremont et al., 2004). For neural-network compression, quantized sparse weight decomposition combines structural constraints, while activation-aware and truncation-aware SVD methods use calibration statistics to preserve the action of a weight matrix on likely inputs (Kuzmin et al., 2022; Yuan et al., 2023; Wang et al., 2025). NaNA uses rank-one components from an exact SVD as intervention units (Xue & Andrzejak, 2026). In an exact SVD, however, the left and right singular vectors are generally dense, so each component connects to most input and output dimensions. Double Sparse Factorization (DSF) instead approximates a dense weight matrix as the product of two sparse matrices (Boza & Macko, 2025). SWD adopts DSF and uses calibration inputs when fitting its two sparse matrices so that the replacement preserves the outputs of the original dense matrix. The exact-SVD controls in Section 3.5 apply the same attribution and ablation protocol to dense SVD components and sparse SWD components, testing whether sparse read/write connections reduce the number of selected units and active edges required to reach the same sufficiency and necessity levels.

C Limitations

This appendix expands the limitations summarized in Section 5.

Efficient extraction is not complete understanding. SWD reaches matched replacement fidelity while using less than 1% of the data that trained baselines use to train their replacements, and it exposes circuits with fewer active edges. These circuits, however, remain local, task-specific objects rather than a global account of the model. This local-to-global gap is shared by other MI approaches: learned feature dictionaries and task-circuit methods provide tractable local objects without, by themselves, yielding a complete account of the model (Cunningham et al., 2023; Conmy et al., 2023; Syed et al., 2023; Ameisen et al., 2025). Thus, our efficiency gains do not resolve the broader concern that a large nonlinear model may not admit a compact, human-understandable mechanistic account (Hendrycks & Hiscott, 2025). Our sufficiency and necessity results establish that selected bottleneck units causally affect held-out task margins, but they do not explain the model globally or guarantee coverage of rare, safety-relevant cases.

SWD circuits are local and non-unique. Identifiability of a two-factor sparse matrix decomposition requires additional structural conditions and is, even then, only defined up to unavoidable scaling and permutation symmetries (Zheng et al., 2021). SWD does not establish such conditions for its approximate learned decompositions, so the resulting bottleneck units can depend on the objective, calibration distribution, initialization, and sparsity budget. Moreover, SWD generally intervenes on an approximate replacement $\boldsymbol{AB}$: matched cross-entropy and reconstruction control average drift but cannot guarantee agreement with the original dense model on every input. First-order ranking can miss behaviorally important edges, and mean-ablation conclusions can vary with the intervention and evaluation protocol (Hanna et al., 2024; Zhang & Nanda, 2024; Miller et al., 2024); redundant or nonlinear mechanisms are also suggested by Appendix M. Recent controlled evidence further shows that exact circuit claims can change with the reported graph, pruning threshold, query/key representation, and comparison granularity, even when coarser summaries remain stable (Sheng & Fu, 2026). Our comparisons hold the task data, ranking rule, ablation policy, top-k evaluation, and cost accounting fixed across methods; this supports controlled comparisons within our chosen protocol but does not establish invariance to alternative extraction or reporting choices. The extracted circuits should therefore be interpreted as task- and protocol-specific causal explanations, not unique or complete mechanistic ground truth.

Scaling verification remains open. Our Qwen3.5-27B experiment covers one matrix, full-model replacement is evaluated only on GPT-2 Small, and the semantic audit examines six model-labeled bottleneck units. Thus, reducing factorization data and circuit size does not remove the need to verify hypotheses across decompositions, prompt distributions, and adversarial edge cases. Following the top-down alternative emphasized by Hendrycks & Hiscott (2025), SWD is best used alongside behavioral and representation-level analyses rather than as a certificate of understanding or safety.

D EXPERIMENTAL DETAILS

This section consolidates the settings shared across experiments. We first specify the replacement surfaces, then give the SWD optimization, circuit-selection protocol, task data, and baseline configurations. Throughout our experiments, replacement fidelity is measured on data excluded from replacement fitting. For circuit comparisons, unit scores and mean-ablation values are estimated on `circuit_train`, and the resulting nested top-k prefixes are evaluated on `circuit_test`.

D.1 MODELS AND REPLACEMENT TARGETS

Table 2 summarizes the computational surfaces used in the paper. Within each single-matrix comparison, all methods replace the same weight matrix from the pretrained model. The MLP-replacement experiment is reported separately because a standard Transcoder replaces the entire MLP, whereas the full-model experiment replaces all attention and MLP weight matrices across the transformer blocks and therefore tests accumulated approximation error.

Table 2: Models and replacement surfaces. Each comparison uses a common surface across the methods shown in that comparison.

Setting	Model	Replacement surface
GPT-2 single matrix	GPT-2 Small	Layer 8 `mlp.c_proj`
Qwen2.5 scaling	Qwen2.5 0.5B, 1.5B, 3B	Middle-layer `mlp.down_proj`: layers 12, 14, and 18, respectively
Qwen3.5 single matrix	Qwen3.5-27B	Layer 31 `mlp.down_proj`
GPT-2 MLP replacement	GPT-2 Small	Layer 8 standard MLP, comprising `mlp.c_fc` and `mlp.c_proj`
GPT-2 full-model replacement	GPT-2 Small	`attn.c_attn`, `attn.c_proj`, `mlp.c_fc`, and `mlp.c_proj` in all 12 blocks
Zero-data	GPT-2 Small	Layer 8 `mlp.c_proj`, matching the GPT-2 single-matrix experiment
Qualitative audit	GPT-2 Small	`mlp.c_proj` in all 12 blocks; selected bottleneck units are visualized from layers 6, 8, and 10

D.2 SWD FACTORIZATION AND FIXED-SUPPORT FINE-TUNING

We implement SWD with the activation-aware Double Sparse Factorization solver of Boza & Macko (2025). Unless stated otherwise, decompositions use the standard inner width $m = \min(d_{\text{in}}, d_{\text{out}})$; the only exception is the overcomplete `mlp.c_fc` factorization in Appendix M. The following configuration is fixed across a sparsity sweep; only the total nonzero budget K changes.

- **Budget and initialization.** The square factor receives density 0.16 for square targets and 0.25 for rectangular targets, with the remaining entries in K assigned to the other factor. We rescale the target using the diagonal of the calibration Gram matrix, initialize the square factor to the identity and the other factor to a magnitude-pruned rescaled target, and initialize the dual variables to zero.
- **Alternating optimization.** Each outer iteration updates both factors in turn. We use 40 outer iterations for the GPT-2 single-matrix and full-model experiments and 8 for the Qwen single-matrix experiments.
- **ADMM block updates.** Each block update consists of a regularized least-squares solve, hard thresholding to enforce the prescribed nonzero budget, and a dual-variable update. We use 5 inner iterations, penalty $\rho = 1$, and ridge coefficient 10^{-2}. Hard thresholding recomputes the prescribed support during the first 2 inner iterations and then holds it fixed. At outer iteration t of T, the first solve uses $\rho_{\text{init}} = \min\{1, t/(T-3)\}^3$; subsequent inner solves use $\rho = 1$.
- **Finalization and precision.** After alternating optimization, 20 fixed-support ADMM iterations refit the surviving entries of $\boldsymbol{B}$ while holding $\boldsymbol{A}$ and both factor supports fixed. Factorization

uses float32 with TF32 matrix multiplication disabled. This local finalization is distinct from the model-level fixed-support fine-tuning used in the full-model experiment below.

- **Data used to fit SWD replacements.** Each point on an SWD replacement-quality curve corresponds to an independently fitted SWD replacement using the stated amount of calibration data; the points are not successive checkpoints from one training run. Table 3 separates the data ranges evaluated in these curves from the fixed SWD replacements subsequently used for circuit extraction.

Table 3: Data used for SWD replacement-quality results and the fixed SWD replacements used for circuit extraction. CE delta is replacement CE minus dense-model CE. The third column identifies the fixed replacement whose units are subsequently scored and ablated; it does not count the task data used for circuit scoring and evaluation.

Experiment	CE-curve data (tokens)	Fixed replacement used for circuit extraction
GPT-2 single matrix	1,024–4.19M	s=0.5: 16,384 tokens; CE delta 0.000889 s=0.75: 16,384 tokens; CE delta 0.008292
Qwen2.5-0.5B	1,024–33.55M	s=0.5: 1,024 tokens; CE delta 0.001222 s=0.75: 1.05M tokens; CE delta 0.005010
Qwen2.5-1.5B	1,024–33.55M	s=0.5: 1,024 tokens; CE delta 0.000733 s=0.75: 1.05M tokens; CE delta 0.001222
Qwen2.5-3B	1,024–33.55M	s=0.5: 2,048 tokens; CE delta 0.000733 s=0.75: 1.05M tokens; CE delta 0.000611
Qwen3.5-27B	1,024–16.78M	s=0.5: 2,048 tokens; CE delta -0.000427 s=0.75: 1.05M tokens; CE delta -0.000448
GPT-2 MLP replacement	1,024–33.55M	s=0.5: 16,384 tokens; CE delta 0.004003 s=0.75: 16,384 tokens; CE delta 0.020491
GPT-2 full model	4.19M (factorization) + 16.38M (fine-tuning)	Final SWD-FT replacement (20.57M total tokens)
Zero-data SWD	0	Fixed zero-data factors at each evaluated sparsity

In the primary GPT-2 and Qwen2.5 single-matrix comparisons, $s = 0.5$ is the CE-matched SWD setting and $s = 0.75$ is a higher-sparsity reference. Both fixed SWD settings are included in the Qwen3.5-27B circuit evaluation. In the MLP-replacement experiment, $s = 0.5$ is the CE-matched SWD setting and $s = 0.75$ is a higher-sparsity reference; SWD composes the two projection factorizations as specified in Appendix G. The full-model experiment instead compares methods at a matched active-weight budget, and zero-data SWD uses no calibration data (Appendix I.1).

Circuit cost. Each selected unit is one SWD bottleneck unit. Its read and write vectors are $\boldsymbol{A}_{:,i}$ and $\boldsymbol{B}_{i,:}$, so its active-edge count is $\|\boldsymbol{A}_{:,i}\|_0 + \|\boldsymbol{B}_{i,:}\|_0$.

Fixed-support fine-tuning. For the full-model experiment, approximation errors from the 48 replaced weight matrices accumulate. We therefore apply model-level fixed-support fine-tuning after sparse factorization. All pretrained parameters and binary factor masks remain frozen, and gradients update only factor values at existing nonzero locations. Fine-tuning thus preserves both the 26,770,800-nonzero budget and the read/write connectivity used for circuit-cost accounting.

- **Objective and data.** Fixed-support fine-tuning minimizes autoregressive next-token cross-entropy on 4.19M unique training tokens. We train with batch size 8 for 2,000 steps, totaling 16.38M consumed tokens.
- **Optimizer.** We use AdamW with peak learning rate 5×10^{-7}, $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 0.1$, and weight decay 10^{-3}.
- **Schedule and gradient control.** The learning rate warms up linearly over the first 1% of steps and then decays linearly to zero. Before each optimizer step, the global factor-gradient RMS is normalized to 1.0.
- **Precision and seed.** Fine-tuning uses bfloat16 training, float32 evaluation, and seed 0.

D.3 Unit Scoring and Circuit Selection

Candidate units are scored under the unpruned replacement. SWD exposes bottleneck units, Transcoders expose hidden features, and the VPD variants expose parameter components. For the sparse-pretrained model, which has no explicit replacement bottleneck, we prune latent channels following its native setup (Gao et al., 2025).

Let $g_\tau(M, \boldsymbol{x})$ be the task logit margin for example $\boldsymbol{x}$, computed from the next-token logits as the average correct-answer logit minus the average distractor-answer logit, and let $z_i(\boldsymbol{x})$ be the activation of candidate unit i (for SWD, $z_i(\boldsymbol{x}) = \boldsymbol{h}_{\ell,t}(\boldsymbol{x})\boldsymbol{A}_{:,i}$ at the task-scoring position). The train-split mean activation used for ablation is

$$\mu_i = \mathbb{E}_{\boldsymbol{x}\in\mathcal{D}_{\tau,\text{train}}}\left[z_i(\boldsymbol{x})\right].$$

We score units by positive first-order task-margin attribution relative to the value used for ablation (Bushnaq et al., 2026; Syed et al., 2023),

$$a_i = \mathbb{E}_{\boldsymbol{x}\in\mathcal{D}_{\tau,\text{train}}}\left[\max\left(0,\ (z_i(\boldsymbol{x}) - r_i)\frac{\partial g_\tau}{\partial z_i(\boldsymbol{x})}\right)\right],$$

where r_i is the value that replaces z_i when unit i is ablated. The main circuit comparisons use mean ablation, so $r_i = \mu_i$; the zero-ablation analysis (Appendix E.3) and the qualitative study use $r_i = 0$. Here $\partial g_\tau / \partial z_i$ is the gradient of the task margin with respect to unit i's activation, so a_i accumulates only positive first-order contributions to the margin. We sort each method's candidate units once by a_i on `circuit_train`; this fixed ranking defines the nested top-k prefixes evaluated on held-out `circuit_test`.

Given the average task score $Q_\tau(M;\mathcal{D}) = \mathbb{E}_{\boldsymbol{x}\in\mathcal{D}}[g_\tau(M,\boldsymbol{x})]$ and M_{unpruned} denoting the replacement before circuit pruning, we report method-relative sufficiency,

$$\text{Suff}_\tau(S;\mathcal{D}) = \frac{Q_\tau(M_{\text{keep}}(S);\mathcal{D})}{Q_\tau(M_{\text{unpruned}};\mathcal{D})},$$

and

$$\text{NecDrop}_\tau(S;\mathcal{D}) = Q_\tau(M_{\text{unpruned}};\mathcal{D}) - Q_\tau(M_{\text{abl}}(S);\mathcal{D}).$$

For a selected set S, let $\mathbf{r}_i$ and $\mathbf{w}_i$ denote the read and write weight vectors of unit i, and let $S_{\text{eff}} \subseteq S$ be its valid subset (Appendix A). We measure circuit size by

$$C_{\text{unit}}(S) = |S|, \qquad C_{\text{edge}}(S) = \sum_{i\in S_{\text{eff}}} \left(\|\mathbf{r}_i\|_0 + \|\mathbf{w}_i\|_0\right).$$

Method-specific definitions of $\mathbf{r}_i$ and $\mathbf{w}_i$ are given in Appendices D.2 and D.5.

D.4 Data Sources and Splits

D.4.1 Language-Model Data

In the replacement and circuit comparisons, all data-dependent replacement methods use FineWeb-Edu for fitting or training (Lozhkov et al., 2024). The qualitative analyses in Appendices K–M use separately fitted factorizations whose settings are stated there. SWD uses subsets of this corpus to fit its sparse factors, whereas the trained baselines use it for optimization. Replacement quality is evaluated on a fixed, disjoint FineWeb-Edu split shared by all methods within each comparison. Accordingly, data used counts calibration tokens for SWD and optimizer-consumed tokens for the trained baselines. Method-specific data amounts and the checkpoints used for circuit extraction are reported in Appendices D.2 and D.5. The zero-data experiment uses no text data.

D.4.2 Circuit-Task Data

Circuit extraction uses fixed public examples from Edge-Pruning (Bhaskar et al., 2024) and auto-circuit (Conmy et al., 2023; Miller et al., 2024). These task examples are separate from the FineWeb-Edu data above and are not counted as replacement fitting or training data. The training split is used to estimate unit scores and mean-ablation values; the test split is used only to evaluate the resulting fixed top-k sets.

Table 4: Public circuit-task sources and train/test example counts. The linked repository revisions fix the source data used in our experiments.

Task	Public source	GPT-2	Qwen2.5	Qwen3.5-27B
Greater-than	Edge-Pruning, `a918efd8`	80,000 / 12,240	Not used	Not used
IOI	auto-circuit, `8fc5afd8`	600 / 200	600 / 200	600 / 200
Docstring	auto-circuit, `8fc5afd8`	600 / 200	600 / 200	600 / 200
Gendered pronoun	Edge-Pruning, `a918efd8`	3,024 / 378	3,024 / 378	2,887 / 370

GPT-2 uses all four tasks. Qwen2.5 and Qwen3.5-27B use IOI, docstring, and gendered pronoun. Greater-than is omitted for Qwen because its numerical answers do not satisfy the single-token next-token evaluation used in these experiments. The Qwen3.5-27B counts report the examples retained after model-specific tokenization compatibility checks.

D.5 Baseline Configurations

Shared data-usage axes count optimizer-replay tokens for trained baselines and calibration tokens for SWD. They compare data used and do not equate optimization steps or FLOPs. We match replacement CE to within 0.001 unless a curve is explicitly included as a reference with its CE delta reported.

Table 5 summarizes the representation, sparsity, optimization, data requirements, and scope of each method. The subsections below provide the exact configurations and checkpoints used in each experiment.

D.5.1 Transcoder

The Transcoder baselines use sparse hidden-feature replacements for the same target surface as the corresponding SWD comparison. Below we report the settings needed to identify the plotted Transcoder lines and the checkpoints used for circuit extraction.

Circuit cost. Each selected unit is one Transcoder hidden feature. Its active edges are the nonzero encoder and decoder weights connected to that feature.

GPT-2 single-matrix replacement. For the GPT-2 single-matrix experiment, the Transcoder replaces the layer 8 MLP output projection.

- **Target:** GPT-2 Small layer 8 `mlp.c_proj`.
- **Widths:** TC3k: Transcoder with hidden width 3072, TC12k: Transcoder with hidden width 12,288.
- **Feature sparsity penalty:** L1 coefficient 5×10^{-5}, seed 42.
- **Replacement-quality curve:** measured from 1,024 to 4.19M training tokens.
- **Checkpoint for circuit extraction:** 2.048M training tokens; CE deltas are 0.000529 for TC3k and 0.000979 for TC12k.

Qwen2.5 down-projection scaling. For the Qwen2.5 scaling experiment, each Transcoder replaces the middle-layer MLP down-projection of the corresponding model size: layer 12 for Qwen2.5-0.5B, layer 14 for Qwen2.5-1.5B, and layer 18 for Qwen2.5-3B.

- **Widths:** TC3k and TC12k, with hidden widths 3,072 and 12,288.
- **Feature sparsity penalty:** L1 coefficient 5×10^{-5}, seed 42.
- **Replacement-quality curve:** measured from 1,024 to 33.55M training tokens.
- **Checkpoints for circuit extraction:** Qwen2.5-0.5B uses 25.17M training tokens for both widths, with CE deltas 0.001654 for TC3k and 0.002032 for TC12k. Qwen2.5-1.5B uses 33.55M tokens for TC3k and 25.17M for TC12k, with CE deltas -0.000057 and -0.000098, respectively. Qwen2.5-3B uses 25.17M tokens for both widths, with CE deltas 0.001516 for TC3k and 0.001229 for TC12k.

Table 5: Comparison of the sparse intervention methods evaluated in this paper.

Property	**SWD (ours)**	**Transcoder**	**VPD**	**Sparse pretrain**
Separate model training (additional optimization cost)	**No**; fit sparse factors **post hoc** (the full-model setting additionally fine-tunes existing nonzero values)	**Yes**; train a replacement network for the pretrained model	**Yes**; train a decomposition and input-dependent selector for the pretrained model	**Yes**; train the complete sparse model **from scratch**
Representation and unit of analysis	Two sparse factors; one bottleneck unit is a **rank-one read–write path**	A replacement network; one unit is a **hidden feature**	A parameter decomposition; one unit is a **rank-one parameter component**	A sparse pretrained model; one unit is a native **hidden channel**
Type of sparsity	**Weight sparsity** in the factor matrices	**Activation sparsity** in hidden features	**Input-dependent component selection** over parameter components	**Weight sparsity** throughout the model; activation sparsity is disabled in our run
Optimization objective	Reconstruct target weights or their outputs on calibration data under fixed factor sparsity	Reconstruct original MLP outputs while encouraging sparse feature activations	Match model outputs or replaced-matrix outputs, with parameter reconstruction and sparse component use	Minimize language-model loss while progressively sparsifying weights
Data required	**None or a small amount** of calibration data*	Text used to train the replacement network	Text used to train the decomposition and selector	**Full pretraining corpus**
Applicable scope	MLP and attention matrices; from one matrix to all model projections	MLP computations; one MLP at a time in our comparisons	MLP and attention matrices; from one matrix up to full-model replacement	Whole model during pretraining

*At matched replacement CE, SWD uses **less than 1%** of the data used by the corresponding trained baseline in our experiments. The baselines' greater data use does not translate into lower circuit cost: SWD typically reaches the same sufficiency or necessity target with fewer active edges; see Sections 3.3 and 3.4.

Qwen3.5-27B down projection. For Qwen3.5-27B, each Transcoder replaces layer 31 `mlp.down_proj`.

- **Widths:** TC24k and TC48k, with hidden widths 24,576 and 49,152.
- **Feature sparsity penalty:** L1 coefficient 5×10^{-5}, seed 42.
- **Replacement-quality curves:** measured from 1,024 tokens; TC24k is shown through 134.22M training tokens and TC48k through 67.11M training tokens.
- **Checkpoints for circuit extraction:** 134.22M training tokens for TC24k and 67.11M training tokens for TC48k, with CE deltas 0.000398 and 0.000695, respectively.

GPT-2 MLP replacement. For the MLP-replacement comparison, we use the standard TC24k Transcoder setting with hidden width 24,576, L1 coefficient 1.4×10^{-4}, and seed 42. Circuit extraction uses only the 6.29M-token checkpoint, with CE delta 0.003456. The complete replacement-quality trajectory is reported in Appendix G.

D.5.2 VPD-KL AND VPD-RECON-CI

VPD-KL uses the original VPD component parameterization, learned causal-importance (CI) mask, and model-level KL fidelity term. VPD-Recon-CI retains the same parameterization and mask but

replaces this term with local activation reconstruction. The differing fidelity terms are

$$\mathcal{L}_{\text{fid}}^{\text{VPD-KL}} = \mathbb{E}_{\boldsymbol{x},t}[D_{\text{KL}}(p_{\text{dense},t}(\cdot \mid \boldsymbol{x}) \,\|\, p_{\text{VPD-KL},t}(\cdot \mid \boldsymbol{x}))]\,,$$
$$\mathcal{L}_{\text{fid}}^{\text{VPD-Recon-CI}} = \mathbb{E}_{\boldsymbol{x},t,j}\left[\left(\widehat{y}_{\ell,t,j}^{\text{CI}}(\boldsymbol{x}) - y_{\ell,t,j}(\boldsymbol{x})\right)^2\right].$$

Here $\boldsymbol{x}$ denotes a token sequence, t a next-token position, ℓ the target module, and j one coordinate of that module's output. The distributions $p_{\text{dense},t}$ and $p_{\text{VPD-KL},t}$ are the dense and VPD-KL next-token distributions. The values $y_{\ell,t,j}$ and $\widehat{y}_{\ell,t,j}^{\text{CI}}$ are the corresponding dense and CI-masked replacement outputs; the expectation in the reconstruction objective averages over all batch elements, token positions, and output coordinates, matching the elementwise MSE used in training. The first objective constrains the final model output; the second directly constrains the local replacement. Because VPD-KL does not constrain the replaced projection directly, its replacement CE cannot always be matched, and it is then reported only as a replacement-quality reference. The equations isolate the fidelity terms; mask-minimality regularization is specified below. The CI network predicts a per-component mask, after which the masked rank-one components are recombined into the replacement output.

Circuit cost. Each selected unit is one VPD parameter component. Its active edges are the nonzero entries in that component's read and write vectors. The CI predictor is not part of this read/write edge count.

GPT-2 single-matrix replacement. For the GPT-2 single-matrix experiment, VPD-Recon-CI replaces the layer 8 MLP output projection.

- **Target:** GPT-2 Small layer 8 `mlp.c_proj`.
- **Components and CI predictor:** $C = 3{,}072$ components, seed 0, and a vector-MLP CI predictor of hidden width 128.
- **Objective:** MSE between the CI-masked replacement and dense projection outputs, plus a CI minimality penalty of 10^{-4}.
- **Checkpoint for replacement quality and circuit extraction:** 4.19M training tokens, with CE delta 0.000873. We reuse this fixed checkpoint for both replacement-quality and circuit results.
- **VPD-KL reference:** the native model-level KL objective on the same target. Its replacement-quality curve is measured from 4,096 to 1.6384B training tokens; it is not used for circuit extraction (Appendix E.1).

Qwen2.5 down-projection scaling. For the Qwen2.5 scaling experiment, VPD-Recon-CI replaces the same middle-layer MLP down-projection used by SWD and the Transcoders.

- **Targets:** layer 12 `mlp.down_proj` for Qwen2.5-0.5B, layer 14 for Qwen2.5-1.5B, and layer 18 for Qwen2.5-3B.
- **Components and objective:** $C = 12{,}288$ components and the same local CI-masked reconstruction objective as in the GPT-2 single-matrix experiment.
- **Checkpoints for circuit extraction:** 16.78M training tokens for Qwen2.5-0.5B, 4.19M for Qwen2.5-1.5B, and 16.78M for Qwen2.5-3B. Their CE deltas are 0.000729, 0.001186, and 0.000906, respectively. The corresponding circuit results are reported in Appendix E.4.

Qwen3.5-27B down projection. For Qwen3.5-27B, VPD-Recon-CI replaces layer 31 `mlp.down_proj`.

- **Components and CI predictor:** $C = 12{,}288$ components, seed 0, and a vector-MLP CI predictor of hidden width 12.
- **Objective:** MSE between the CI-masked replacement and dense projection outputs, plus a CI minimality coefficient of 3×10^{-4}.
- **Checkpoint for circuit extraction:** 200.00M training tokens, with CE delta 0.001847. It is retained as the parameter-decomposition reference; the corresponding circuit results are reported in Appendix E.5.

GPT-2 MLP replacement. For the MLP-replacement experiment, VPD-Recon-CI replaces the complete layer 8 MLP.

- **Components and objective:** $C = 3{,}072$ components with the local MLP-reconstruction objective.
- **Replacement-quality curve:** measured from 1,024 to 16.78M training tokens.
- **Checkpoint for circuit extraction:** 4.19M training tokens, with CE delta 0.003522.
- **VPD-KL reference:** $C = 3{,}072$, seed 0, and the native model-level KL objective. Its replacement-quality curve is measured from 4,096 to 1B training tokens; it is not used for circuit extraction. Full results are reported in Appendix G.

D.5.3 Sparse-Pretrained Reference

The sparse-pretraining reference is used in the GPT-2 Small full-model comparison. Unlike the other baselines, it is a separately trained sparse model with GPT-2 Small dimensions, rather than a sparse replacement of the Hugging Face GPT-2 checkpoint.

Circuit cost. For circuit extraction, the 3,072 post-GELU MLP channels in each layer form 36,864 candidate units; attention and residual channels are not candidate units. Each selected unit is one MLP hidden channel. Its active edges are the nonzero incoming and outgoing weights connected to that channel. For channel j, this count is

$$\|\boldsymbol{W}_{\text{c_fc}}[:, j]\|_0 + \|\boldsymbol{W}_{\text{c_proj}}[j, :]\|_0 + \mathbf{1}[b_{\text{c_fc},j} \neq 0] \,.$$

GPT-2 full-model comparison.

- **Architecture:** 12 layers, residual width 768, 12 attention heads of width 64, MLP width 3,072, context length 1,024, and vocabulary size 50,257. The model uses learned positional embeddings, LayerNorm, tied token-embedding and language-model-head weights, biases, zero dropout, and standard `nn.GELU`. Activation sparsity is disabled. The use of `nn.GELU` differs from the `gelu_new` implementation in the Hugging Face GPT-2 checkpoint used by SWD.
- **Training data:** FineWeb-Edu documents are separated by end-of-sequence tokens and packed into 1,024-token examples; a disjoint FineWeb-Edu split is held out for evaluation. The global batch contains 128 examples, or 131,072 token presentations per optimizer update.
- **Evaluation checkpoints:** The replacement-quality and circuit results use separately trained models with identical configurations, each evaluated after 2,883,584,000 token presentations. Their held-out CE values are 3.450415 and 3.450323, respectively. The complete training schedule contains 4,999,872,512 token presentations, so the evaluated models are not terminal checkpoints.
- **Matched active-weight budget:** At the evaluated data point, the scheduled density is 0.315194486. The 48 transformer-block matrices contain 26,770,920 nonzeros (realized density 0.315194307), compared with 26,770,800 nonzeros in SWD-FT; the difference is 120 weights. This accounting excludes biases, normalization parameters, embeddings, and the tied language-model head. Across all trainable two-dimensional matrices, including token and positional embeddings, 39,184,479 of 124,318,464 entries are nonzero.

Optimization and sparsification. Training uses seed 0, bfloat16, and AdamW with base learning rate 3×10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 0.1$, and weight decay 10^{-3}. Let n denote cumulative token presentations, with $n_0 = 49{,}938{,}432$, $n_1 = 3{,}999{,}793{,}152$, and $N = 4{,}999{,}872{,}512$. The scheduled nonzero density is

$$d(n) = \begin{cases} 1, & n < n_0, \\ 0.2^{(n-n_0)/(n_1-n_0)}, & n_0 \leq n < n_1, \\ 0.2, & n \geq n_1. \end{cases}$$

Thus density remains one during the first 1% of training, decays exponentially to the target density 0.2 by 80% of training, and then remains fixed. The learning rate is

$$\text{lr}(n) = 3 \times 10^{-4} \frac{h(n)}{\sqrt{d(n)}}, \qquad h(n) = \begin{cases} n/n_0, & n < n_0, \\ 1 - (n - n_0)/(N - n_0), & n \geq n_0. \end{cases}$$

After every optimizer update, each trainable two-dimensional matrix is magnitude-projected to its scheduled nonzero budget while retaining at least one weight per output row. The gradient routine computes a global RMS over all available parameter-gradient entries and, when this RMS exceeds 10^{-8}, divides all gradients by $\mathrm{RMS} + 10^{-5}$.

E Single-Matrix Replacement Results

E.1 Replacement Quality beyond Cross-Entropy

Section 3.3 measures replacement quality primarily by cross-entropy delta. We additionally report two complementary metrics on GPT-2 Small layer 8 `mlp.c_proj` and the three Qwen2.5 `mlp.down_proj` surfaces to verify that the comparison is not specific to CE.

Metrics. Let $\boldsymbol{Y}_{\mathrm{dense}}$ and $\boldsymbol{Y}_{\mathrm{rep}}$ stack the dense and replacement outputs of the target projection over all evaluated examples and token positions. The *activation relative MSE* is a local, single-layer measure of how well the replacement reproduces that projection's output,

$$\mathrm{relMSE} = \frac{\|\boldsymbol{Y}_{\mathrm{rep}} - \boldsymbol{Y}_{\mathrm{dense}}\|_F^2}{\|\boldsymbol{Y}_{\mathrm{dense}}\|_F^2}.$$

The numerator and denominator are aggregated over the complete evaluation set before taking the ratio; we do not average per-example relative errors. The *KL* is a global measure that inserts the replacement back into the full model and compares next-token distributions on a held-out language-model split,

$$\mathrm{KL} = \mathbb{E}_t\, \mathrm{KL}\big(p_t^{\mathrm{dense}} \,\|\, p_t^{\mathrm{rep}}\big),$$

with the dense model as the reference distribution and the expectation taken over held-out tokens t. Lower is better for both metrics. The x-axis reports data used in tokens, rather than compute, and uses a broken log scale so the short-horizon comparison and the VPD-KL long tail are both visible. We report activation relative MSE first because it measures the replaced layer directly, and KL second because it measures how that local error propagates to the model's output.

Both metrics reproduce the CE ordering across all single-matrix replacements. SWD reaches low error with little data, whereas Transcoder and VPD-Recon-CI approach it after roughly 10^6 optimizer-replay tokens. VPD-KL is the weakest local replacement because its native objective targets model-level logits instead of the projection output (Appendix D.5.2). No checkpoint on its extended GPT-2 trajectory, from 4,096 to 1.6384B tokens, meets the CE-matching criterion, showing that the fidelity gap is not an early-training artifact.

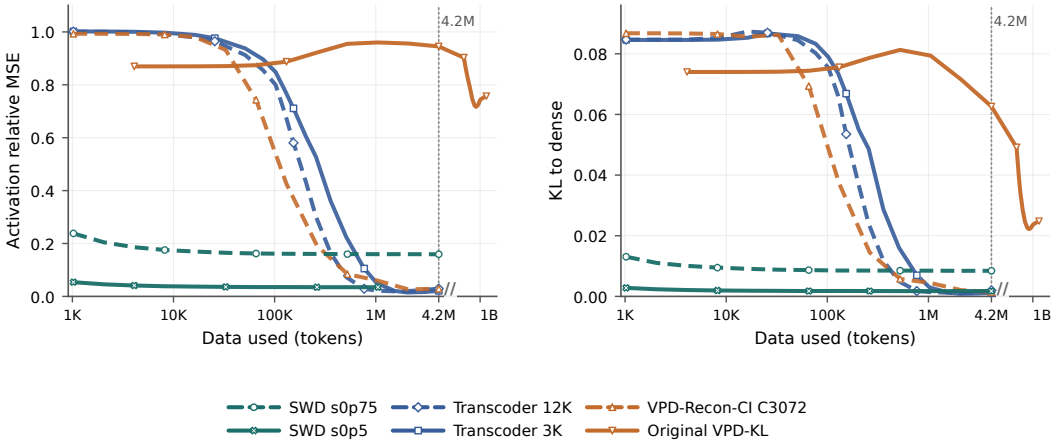

Figure 8: GPT-2 Small layer 8 `mlp.c_proj` single-matrix replacement quality for all methods. Left: activation relative MSE (local). Right: KL to the dense model (global). Lower is better; both panels use the same broken-x policy so the short-horizon comparison and the VPD-KL long tail are visually aligned. VPD-KL is a native-objective replacement-quality reference line and is not a circuit baseline.

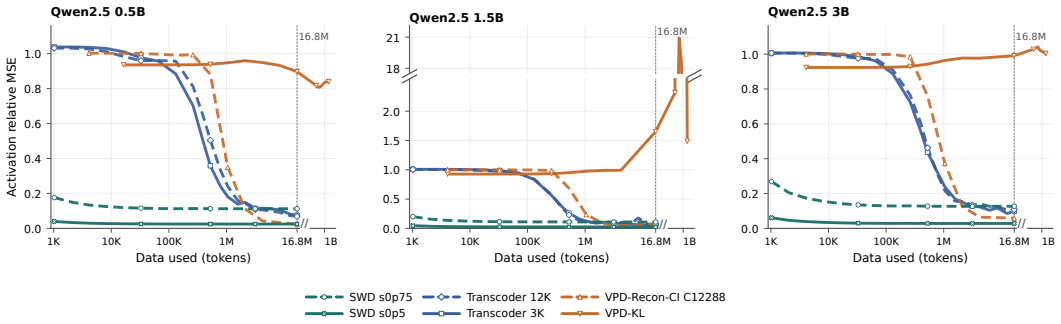

Figure 9: Qwen2.5 0.5B/1.5B/3B single-matrix `mlp.down_proj` activation relative MSE for all methods (lower is better). The panels use the same broken-axis style as the KL companion: 0.5B and 3B use a broken x-axis after the 16.8M-token shared horizon, and 1.5B also uses a y-axis break to keep the low-error region readable despite the VPD-KL long-tail spike. VPD-KL is included to contextualize replacement quality.

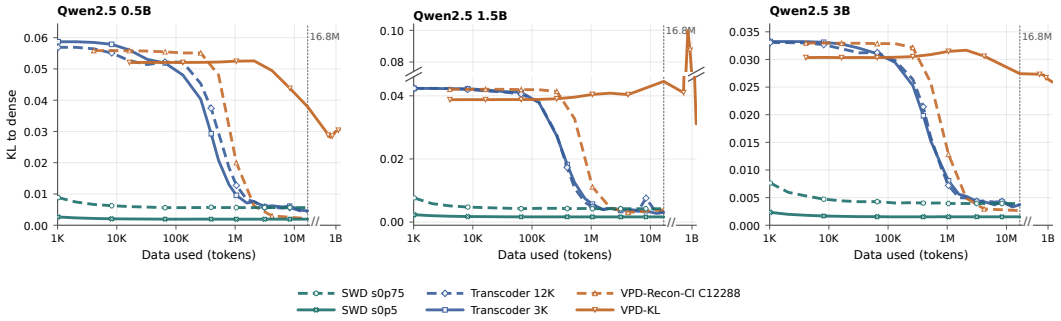

Figure 10: Qwen2.5 0.5B/1.5B/3B single-matrix `mlp.down_proj` KL to the dense model for all methods (lower is better), with the same broken-axis convention as Figure 9.

E.2 GPT-2 Selected-Unit Circuit Results

Figure 11 presents the GPT-2 Small layer 8 `mlp.c_proj` circuit results from Figure 3, using selected units rather than active edges as the cost axis. The main CE-matched comparison uses SWD (s=0.5), TC3k, TC12k, and VPD-Recon-CI; a sparser SWD (s=0.75) is retained as a non-CE-matched sparse control.

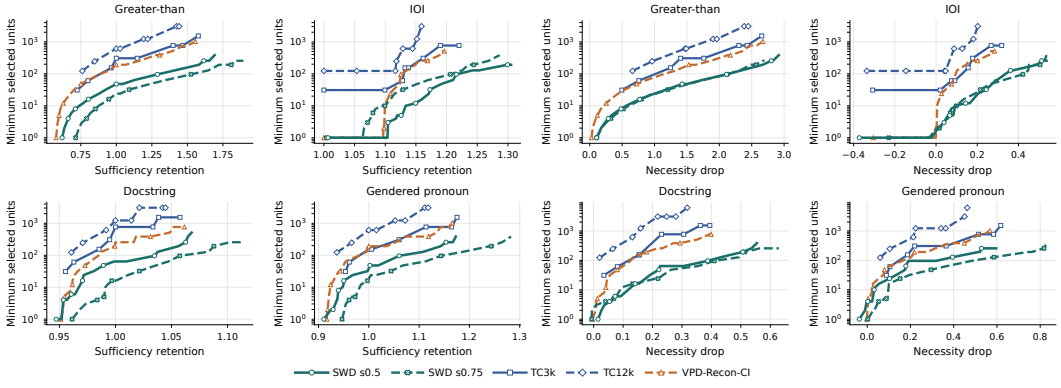

Figure 11: Companion to Figure 3 with selected units as the cost axis. The x-axis is the target threshold and the y-axis is the minimum selected units needed to reach it (lower-right is better). Left: sufficiency. Right: necessity drop. The same SWD advantage holds on the selected-unit axis.

E.3 Robustness under Zero Ablation

The main circuit results replace ablated units with their mean activation on `circuit_train`. To test whether SWD's advantage depends on using mean ablation rather than zero ablation, we repeat the GPT-2 single-matrix circuit evaluation from Section 3.3 using zero ablation. We keep the method checkpoints, task examples, and definitions of selected units and active edges fixed. Consistent with zero ablation, units are ranked on `circuit_train` by a_i with $r_i = 0$ (Appendix D.3). For sufficiency, units outside the selected set are set to zero; for necessity, the selected units are set to zero.

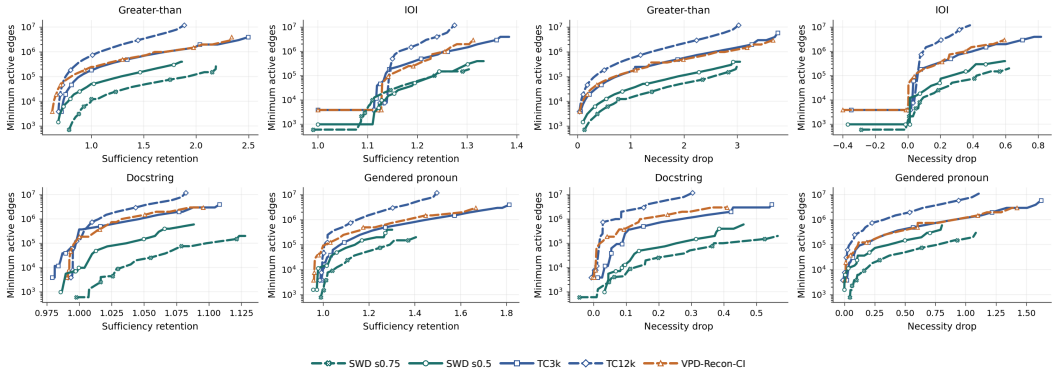

Figure 12: GPT-2 single-matrix circuit results under zero ablation, using minimum active edges as the cost. Left: sufficiency. Right: necessity drop.

Figure 12 shows that SWD's active-edge advantage persists under zero ablation. The CE-matched SWD ($s = 0.5$) reaches most sufficiency thresholds with fewer active edges than TC3k, TC12k, and VPD-Recon-CI on all four tasks, and the higher-sparsity SWD ($s = 0.75$) lowers the cost further; for example, at GreaterThan sufficiency retention of at least 1.0, SWD ($s = 0.75$) requires 12,303 active edges, compared with 184,320 for TC3k. The necessity results show the same overall separation: over the range they reach, the SWD curves attain each necessity drop at lower active-edge cost than the trained baselines, although on IOI their largest necessity drop is smaller than that of TC3k.

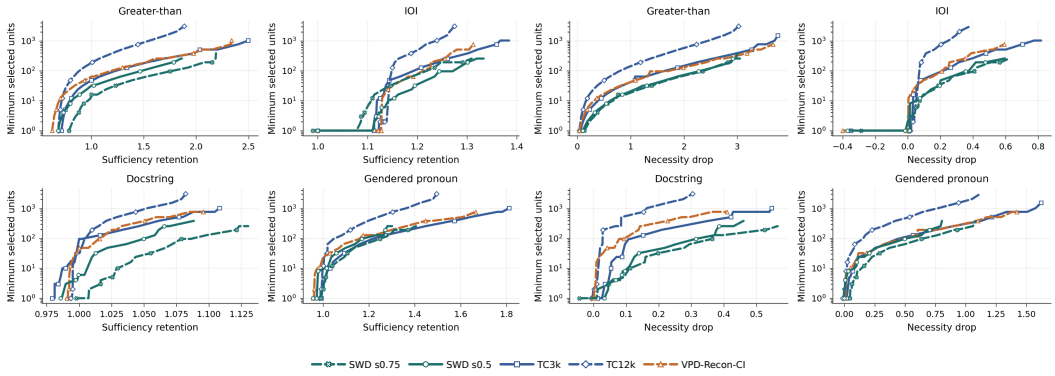

Figure 13: GPT-2 single-matrix circuit results under zero ablation, using minimum selected units as the cost. Left: sufficiency. Right: necessity drop.

The selected-unit comparison in Figure 13 shows the same pattern. Together, these results show that SWD's advantage in the GPT-2 single-matrix experiment is not specific to using mean activation as the ablation reference.

E.4 QWEN2.5 CIRCUIT RESULTS

The Qwen2.5 comparison targets the middle-layer `mlp.down_proj` in each model size. VPD-Recon-CI uses $C = 12{,}288$ components and the same local reconstruction objective as in the GPT-2 experiment. Table 6 specifies the target modules and fixed checkpoints used for circuit extraction. VPD-KL appears only in the replacement-quality figures because its fidelity is insufficient for a controlled circuit comparison. Figure 14 reports replacement quality for all three model sizes. Figures 15–17 report the 0.5B, 1.5B, and 3B circuit results.

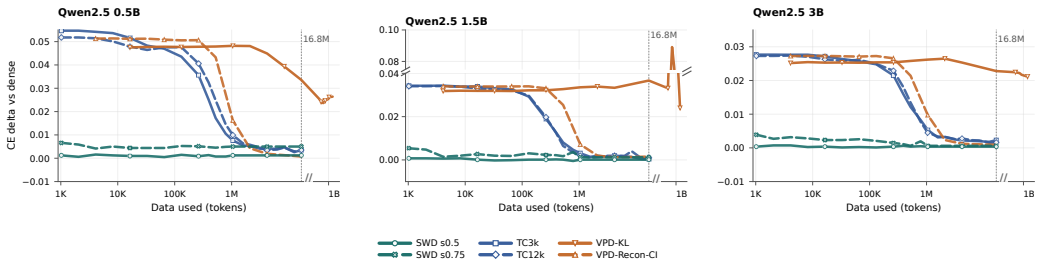

Figure 14: Qwen2.5 single-matrix replacement quality across 0.5B, 1.5B, and 3B models. SWD reaches low CE delta with substantially fewer tokens than the trained baselines.

Table 6: Qwen2.5 VPD-Recon-CI settings. All rows use $C = 12{,}288$ components and replace one `mlp.down_proj`.

Model and target module	Checkpoint for circuit extraction	CE delta
Qwen2.5 0.5B, layer 12 `mlp.down_proj`	16.78M tokens used	0.000729
Qwen2.5 1.5B, layer 14 `mlp.down_proj`	4.19M tokens used	0.001186
Qwen2.5 3B, layer 18 `mlp.down_proj`	16.78M tokens used	0.000906

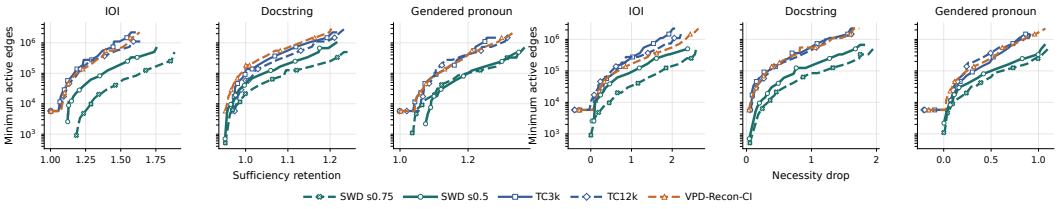

Figure 15: Qwen2.5 0.5B single-matrix circuit results, with active edges as the cost axis (lower-right is better). Left: sufficiency. Right: necessity drop.

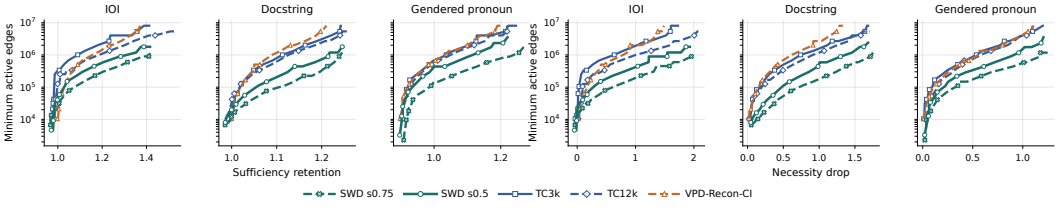

Figure 16: Qwen2.5 1.5B single-matrix circuit results, with active edges as the cost axis (lower-right is better). Left: sufficiency. Right: necessity drop.

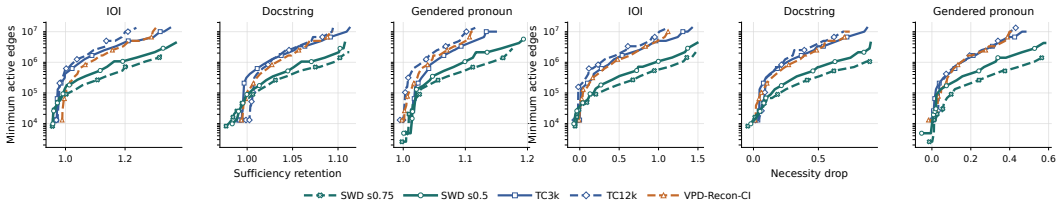

Figure 17: Qwen2.5 3B single-matrix circuit results. SWD reaches the same sufficiency and necessity thresholds with fewer active edges. Left: sufficiency. Right: necessity drop.

Figures 18–20 recast the same three Qwen2.5 circuit results with selected units as the cost axis, separating selected-unit count from read/write connectivity, as in the GPT-2 companion (Figure 11).

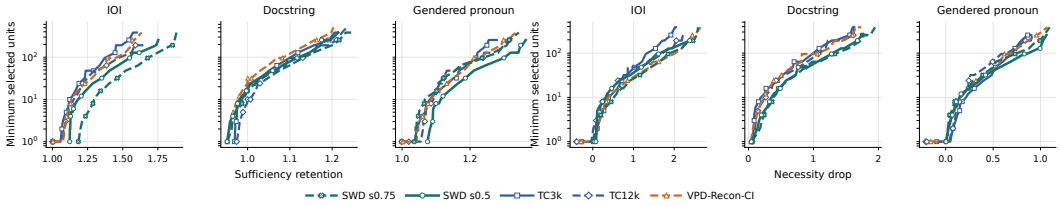

Figure 18: Qwen2.5 0.5B single-matrix circuit results, with selected units as the cost axis (lower-right is better). Companion to Figure 15. Left: sufficiency. Right: necessity drop.

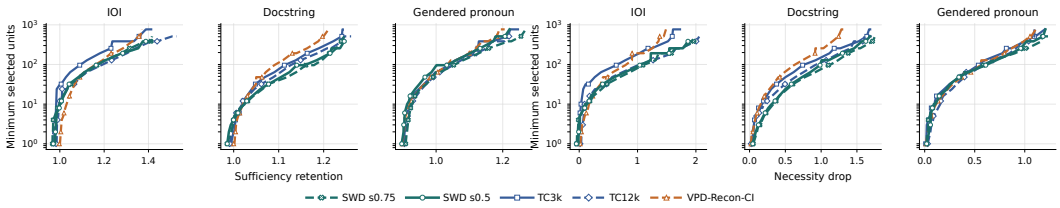

Figure 19: Qwen2.5 1.5B single-matrix circuit results, with selected units as the cost axis (lower-right is better). Companion to Figure 16. Left: sufficiency. Right: necessity drop.

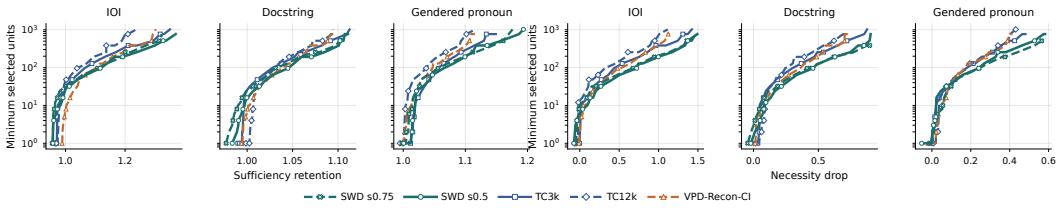

Figure 20: Qwen2.5 3B single-matrix circuit results, with selected units as the cost axis (lower-right is better). Companion to the 3B active-edge results in the main text (Figure 17). Left: sufficiency. Right: necessity drop.

E.5 Qwen3.5-27B Circuit Results

The Qwen3.5-27B comparison targets layer 31 `mlp.down_proj`. All methods use the same IOI, docstring, and gendered-pronoun task data and the circuit-selection protocol in Appendix D.3. Figure 21 reports replacement quality on Qwen2.5-3B and Qwen3.5-27B, Figure 4 reports active edges, and Figure 22 reports the same circuit results using selected units. Table 7 lists the fixed checkpoints used for circuit extraction.

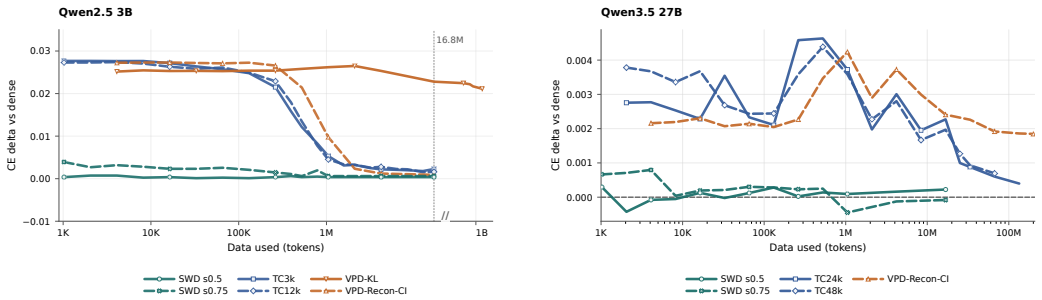

Figure 21: Qwen single-matrix replacement quality. Left: Qwen2.5-3B. Right: Qwen3.5-27B. SWD reaches low CE delta with substantially fewer tokens than the trained baselines. Results for the other Qwen2.5 model sizes are reported in Appendix E.4.

Table 7: Qwen3.5-27B checkpoints used for circuit extraction. CE delta is measured against the same dense model and held-out language-model data.

Method	Setting	Checkpoint for circuit extraction	CE delta
SWD	$s = 0.5$	2,048 calibration tokens	-0.000427
SWD	$s = 0.75$	1.05M calibration tokens	-0.000448
Transcoder	width 24,576	134.22M training tokens	0.000398
Transcoder	width 49,152	67.11M training tokens	0.000695
VPD-Recon-CI	$C = 12{,}288$	200.00M training tokens	0.001847

Relative to SWD with $s = 0.5$, the CE-delta differences are 0.000022 for SWD with $s = 0.75$, 0.000824 for TC24k, 0.001121 for TC48k, and 0.002274 for VPD-Recon-CI. TC48k is slightly outside the 0.001 matching tolerance. VPD-Recon-CI has the largest difference and is retained as a reference for the parameter-decomposition baseline.

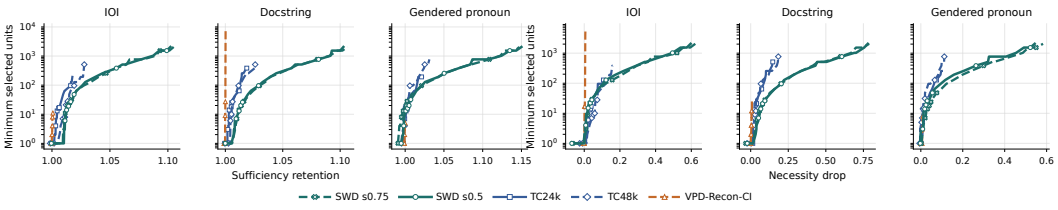

Figure 22: Companion to Figure 4 with selected units as the cost axis. The x-axis is the target threshold and the y-axis is the minimum selected units needed to reach it (lower-right is better). Left: sufficiency. Right: necessity drop.

F Full-Model Replacement

Figure 23 shows the replacement-quality trajectory behind the full-model replacement summary in Section 3.4. The SWD point before fine-tuning appears at 4.19M calibration tokens with CE 3.90. Fixed-support fine-tuning then lowers the CE along the SWD-FT curve without changing the active-weight budget, reaching CE 3.44 at 20.6M total tokens. This matches the sparse-pretraining baseline's CE 3.45, which is reached after 2.884B tokens.

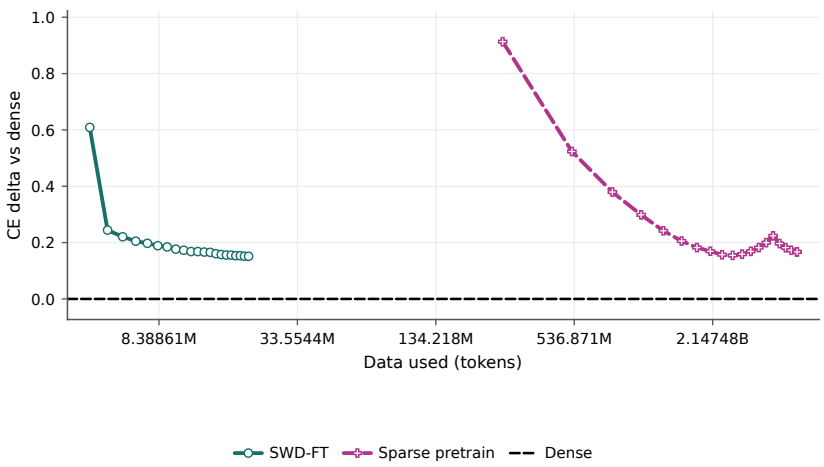

Figure 23: Held-out CE of GPT-2 Small full-model replacements versus data used in tokens (lower is better). The SWD-FT curve begins at the 4.19M-token factorization point and adds fixed-support fine-tuning tokens thereafter; the sparse-pretraining curve counts tokens consumed during sparse pretraining. The dotted horizontal line marks the dense model. At an approximately matched active-weight budget, factorizing and fine-tuning an existing dense checkpoint brings SWD-FT to CE 3.44 after 20.6M tokens, less than 1% of the 2.884B sparse-pretraining tokens used to reach CE 3.45.

G MLP REPLACEMENT

The main single-matrix experiment constrains every method to replace `mlp.c_proj`. Standard Transcoders normally replace the entire MLP, so we also compare on the GPT-2 Small layer 8 MLP input–output map. For SWD, this means factorizing the weight matrices of the two linear projections, `mlp.c_fc` and `mlp.c_proj`, while keeping the original GELU nonlinearity ϕ between them:

$$\widehat{\mathrm{MLP}}(\boldsymbol{h}) = \phi(\boldsymbol{h}\boldsymbol{A}_{\mathrm{fc}}\boldsymbol{B}_{\mathrm{fc}} + \boldsymbol{b}_{\mathrm{fc}})\,\boldsymbol{A}_{\mathrm{proj}}\boldsymbol{B}_{\mathrm{proj}} + \boldsymbol{b}_{\mathrm{proj}}.$$

Thus, SWD changes only the two linear projections, while the nonlinearity is the original GELU used by the dense model.

Table 8: Fixed configurations for the GPT-2 Small layer 8 MLP-replacement comparison. SWD tokens are calibration tokens; Transcoder and VPD-Recon-CI tokens are optimizer replay. VPD-KL is excluded from circuit extraction because its replacement fidelity is not matched.

Method	Replacement configuration	Checkpoint for circuit extraction
SWD	Sequential factorization of `c_fc` and `c_proj`; $s \in \{0.5, 0.75\}$	16,384 tokens used
Transcoder	Standard MLP Transcoder (Appendix D.5.1)	6.29M tokens used; CE delta 0.003456
VPD-Recon-CI	Local MLP-reconstruction objective; $C = 3{,}072$	4.19M tokens used
VPD-KL	Native KL-logit objective; $C = 3{,}072$, seed 0	Replacement only

Replacement-quality trajectories share a 16.78M-token horizon; the VPD-KL curve is extended to 1B tokens to test whether longer optimization closes its fidelity gap. Across CE delta, KL, and activation relative MSE, it remains the weakest replacement. As in the single-matrix experiment, we retain it for replacement-quality context but exclude it from task-circuit pruning.

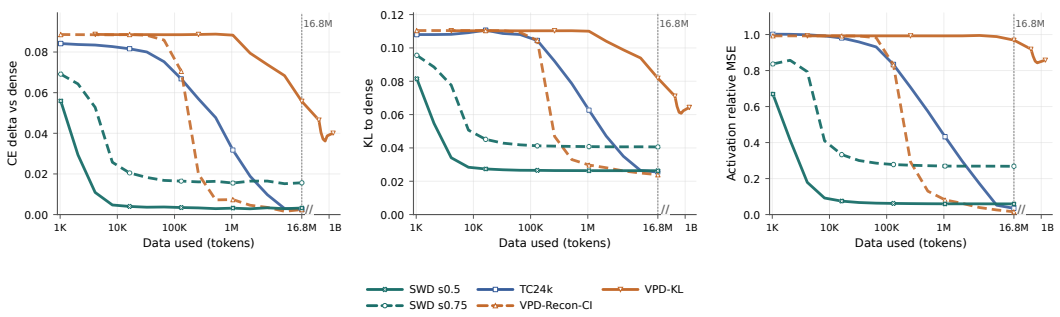

Figure 24: GPT-2 Small layer 8 MLP replacement quality for all methods (lower is better in every panel). Left: CE delta versus the dense model. Middle: KL to the dense model. Right: activation relative MSE at the target module. The broken x-axis keeps the 1K–16.8M shared comparison region readable and shows only the VPD-KL tail beyond the break. The native-objective VPD-KL curve serves as a replacement-quality reference and is excluded from the circuit-pruning figures; it begins at its first 4,096-token checkpoint.

We then apply the same circuit extraction protocol to the fixed MLP-replacement checkpoints in Table 8. The primary matched comparison uses SWD with $s = 0.5$, a single TC24k checkpoint at 6.29M tokens, and the 4.19M-token VPD-Recon-CI checkpoint; SWD with $s = 0.75$ provides a higher-sparsity reference. Figures 25 and 26 report the resulting cost-quality tradeoff using active edges and selected units as the cost axes. On this broader surface, the results are more mixed than in the single-matrix setting: SWD remains competitive on the active-edge axis and often requires fewer edges to reach the same necessity drop, while the standard MLP Transcoder is competitive in several selected-unit and sufficiency comparisons. This comparison complements the single-matrix results by testing the same protocol on the MLP-replacement surface.

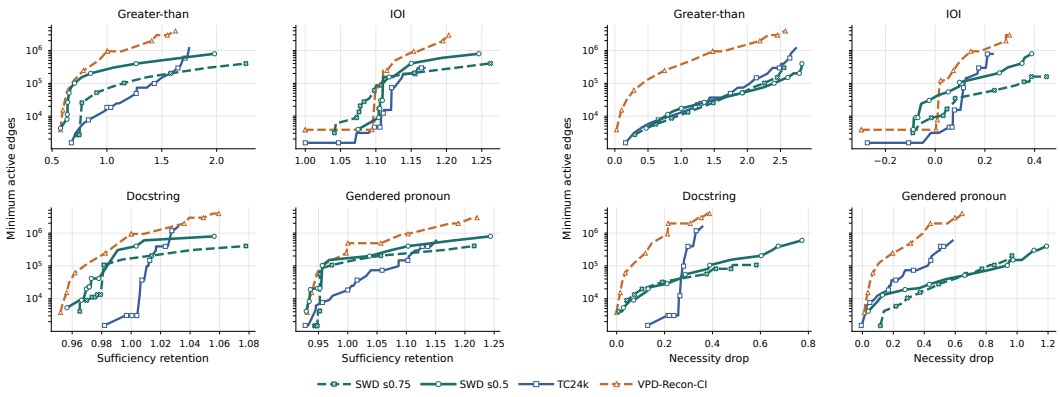

Figure 25: GPT-2 layer 8 MLP-replacement circuit results for the setting in Table 8. For each curve, the x-axis is the target threshold and the y-axis is the minimum active edges needed to reach it (lower-right is better). Left: sufficiency. Right: necessity drop. Transcoder and VPD-Recon-CI use the fixed CE-matched 6.29M- and 4.19M-token checkpoints, respectively.

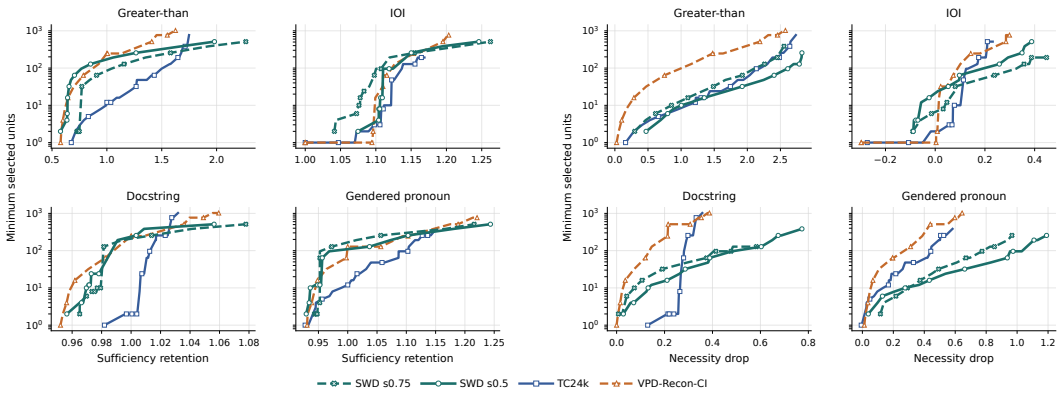

Figure 26: Companion to Figure 25 with selected units as the cost axis (lower-right is better), separating selected-unit count from read/write connectivity.

H Exact-Dense Reparameterization Controls

The exact dense controls in Figure 6 reparameterize GPT-2 Small layer-8 `mlp.c_proj` without approximation or sparsification. They test whether bottleneck re-expression alone yields SWD's low-cost circuits. The SVD control uses the same exact rank-one decomposition used by NaNA (Xue & Andrzejak, 2026). To measure how sparsity in the read and write vectors affects the number of selected units and active edges required, we keep the task data, positive mean-centered first-order task-margin attribution, mean ablation, and figure plotting fixed across SVD, Random-B, and SWD. This shared protocol compares dense and sparse read/write connections without changing how units are ranked or ablated.

Control configurations.

- **SWD ($s = 0.5$):** fixed CE-matched decomposition on the same `mlp.c_proj` surface; the primary matched-fidelity SWD line.
- **SWD ($s = 0.75$):** higher-sparsity SWD reference on the same `mlp.c_proj` surface.
- **SVD Exact:** full-rank dense SVD of `mlp.c_proj` (rank 768, $d_{\text{in}} = 3072$, $d_{\text{out}} = 768$); an exact dense control that is neither low-rank nor sparse.
- **Random-B (seed mean):** random orthogonal exact dense control with $\boldsymbol{B} = \boldsymbol{Q}$, $\boldsymbol{A} = \boldsymbol{W}\boldsymbol{Q}^\top$. Results are averaged over 10 runs (seeds 0–9), and the band shows one standard deviation above and below the mean.

Both dense controls use the same cost accounting: k bottleneck units, or $k(d_{\text{in}} + d_{\text{out}})$ active edges. They exactly reparameterize the GPT-2 layer 8 `mlp.c_proj` matrix, whereas the SWD lines expose sparse read/write structure.

For the full-rank SVD control, the factorization is

$$\boldsymbol{W}_{\text{eff}} = \boldsymbol{A}_{\text{svd}}\boldsymbol{B}_{\text{svd}}, \qquad \boldsymbol{A}_{\text{svd}} = \boldsymbol{U}\boldsymbol{\Sigma}, \quad \boldsymbol{B}_{\text{svd}} = \boldsymbol{V}^\top,$$

using all rank-768 singular directions. For the Random-B control,

$$\boldsymbol{B}_{\text{rand}} = \boldsymbol{Q}, \qquad \boldsymbol{A}_{\text{rand}} = \boldsymbol{W}_{\text{eff}}\boldsymbol{Q}^\top, \qquad \boldsymbol{W}_{\text{eff}} = \boldsymbol{A}_{\text{rand}}\boldsymbol{B}_{\text{rand}},$$

where $\boldsymbol{Q}$ is a seeded random orthogonal matrix from Gaussian QR. In both cases $\boldsymbol{W}_{\text{eff}}$ reproduces the original `mlp.c_proj` map exactly.

Both controls are numerically exact: SVD has mean CE delta 1.58×10^{-6} and KL 1.81×10^{-7}; across 10 Random-B seeds, CE delta ranges from -2.48×10^{-6} to 3.40×10^{-6} and mean KL is 1.89×10^{-7}. Their need for more active edges therefore cannot be attributed to poorer replacement fidelity. On the selected-unit axis (Figure 27), SVD and Random-B are competitive on IOI and at low docstring necessity levels, while SWD requires fewer units in most other regions. The larger and

more consistent difference appears on the active-edge axis, where SWD's sparse read/write factors reduce the number of active connections per selected unit.

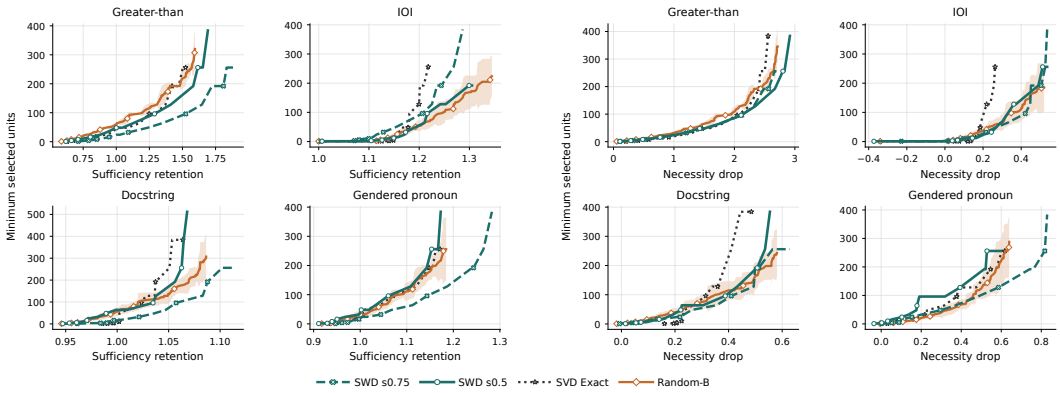

Figure 27: Companion to Figure 6 with selected units as the cost axis (lower-right is better). Left: sufficiency. Right: necessity drop. Dense controls become competitive with SWD at specific low thresholds or tasks, but do not generally reproduce its low-cost circuit.

I ZERO-DATA ANALYSIS

I.1 FACTORIZATION AND EVALUATION SETTING

To isolate the role of calibration activations, we keep the standard SWD solver, factor dimensions, and nonzero-budget schedule fixed and replace only the activation Gram matrix with the identity:

$$\boldsymbol{G} \leftarrow \boldsymbol{I}, \qquad \min_{\boldsymbol{A},\boldsymbol{B}} \|\boldsymbol{W} - \boldsymbol{A}\boldsymbol{B}\|_F^2 \quad \text{s.t.} \quad \|\boldsymbol{A}\|_0 + \|\boldsymbol{B}\|_0 \le K.$$

No calibration activations are used to choose the supports or fit their surviving values. We factorize the GPT-2 Small layer 8 `mlp.c_proj` from the main single-matrix experiment over $s \in \{0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875\}$. Larger s leaves fewer active scalars. Figure 28 compares each resulting product $\boldsymbol{A}\boldsymbol{B}$ with the corresponding activation-calibrated factorization at the same sparsity.

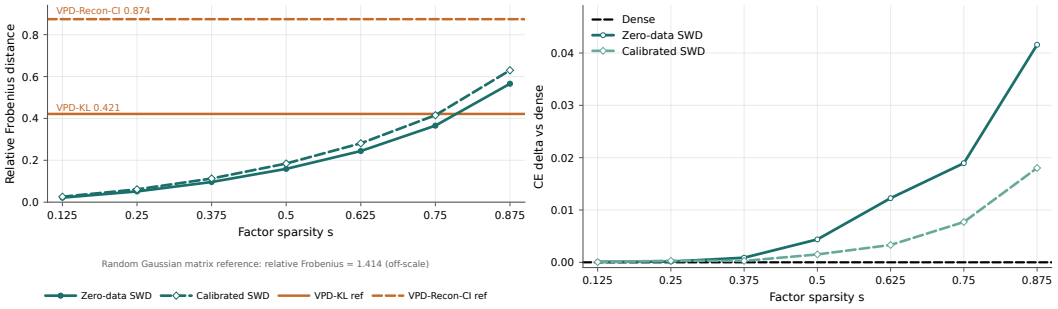

Figure 28: Parameter-space and behavioral fidelity of zero-data and activation-calibrated SWD on the GPT-2 output-projection surface. Left: relative Frobenius reconstruction error. Right: held-out CE increase. Zero-data SWD better preserves the weight matrix, whereas activation calibration better preserves model behavior under aggressive sparsification.

I.2 CIRCUIT RESULTS

Replacement fidelity alone does not establish that the bottleneck units remain useful after circuit pruning. We therefore apply the common unit-scoring and top-k protocol from Section 3.2 to every zero-data sparsity checkpoint. For each of greater-than, IOI, docstring, and gendered-pronoun, the bottleneck-unit ranking and ablation means are computed on `circuit_train`, then frozen before sufficiency and necessity are evaluated on `circuit_test`. Figure 29 reports the minimum active edges and selected units needed to reach each held-out quality threshold. The Transcoder curves use the fixed checkpoints from the single-matrix experiments in Appendix D.5.1 as contextual references.

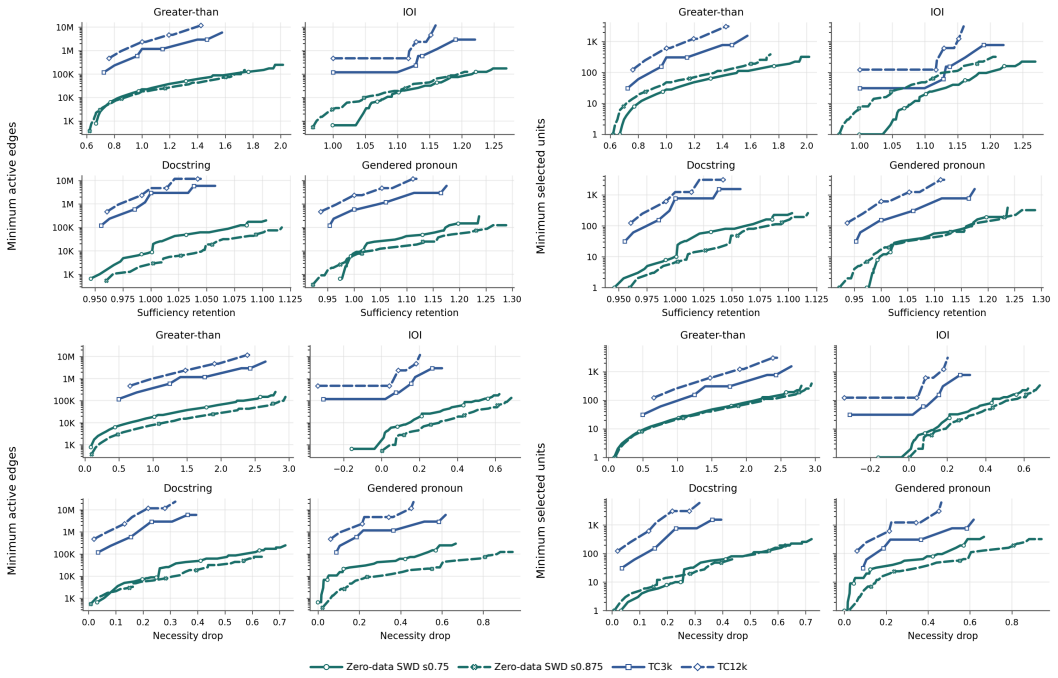

Figure 29: Zero-data identity-Gram SWD task-circuit results compared with fixed Transcoder reference settings. Each curve gives the minimum cost reaching the plotted threshold (lower is better). Top: sufficiency. Bottom: necessity drop. Left: active edges. Right: selected units. Zero-data SWD curves are shown for $s = 0.75$ and $s = 0.875$.

The task results in Figure 29 suggest that zero-data SWD factors are not merely good matrix approximations; their bottleneck units remain useful for circuit extraction after pruning. They also show that many zero-data task-sparsity settings retain high held-out sufficiency at small cost: at $s = 0.75$, the docstring circuit reaches 0.954 test sufficiency with only 2 selected units, and all four task circuits exceed 0.95 with at most 24 selected units.

Useful circuit structure can therefore be recovered directly from pretrained weights without calibration data. Activation-aware SWD retains lower replacement CE under aggressive sparsification (Figure 28), but the zero-data result shows that its bottleneck units are not solely an artifact of a chosen calibration distribution.

J GreaterThan Case Study: Semantic Audit

This appendix provides additional details for the semantic audit in Section 3.7. We rank all 9,208 valid `mlp.c_proj` bottleneck units in GPT-2 Small using the GreaterThan attribution score defined in Appendix D.3. Figure 7 shows the two highest-ranked units in layers 6, 8, and 10, and Table 9 extends the comparison to the top three units in each layer. The global ranks are computed over the full candidate pool.

For each bottleneck unit, we then evaluate 2,048 task-independent WikiText-2 tokens and collect the 20 surrounding text passages associated with its largest activation magnitudes. GPT-5.5 summarizes the recurring patterns in these passages into a short semantic hypothesis. The passages and semantic labels are not used for unit ranking or circuit selection.

Table 9: Top-three GreaterThan bottleneck units in each displayed layer. Layer order is used only for visualization; global rank is over all 9,208 candidates. Semantic hypotheses are task-independent and do not affect ranking.

Layer	Bottleneck unit	Layer order	Global rank	Score	Semantic hypothesis
6	c380	1	108	10.2	measure/unit
6	c723	2	129	9.7	punctuation/function
6	c558	3	152	9.3	mixed lexical
8	c264	1	76	11.3	quantity
8	c511	2	84	11.0	measure/place
8	c728	3	123	9.8	event/time
10	c757	1	32	14.4	numeric/year
10	c627	2	63	11.8	species/name
10	c481	3	85	10.9	numeric/year

The six displayed bottleneck units have global ranks 32–129 and all fall within the global top-512 circuit prefix. Table 10 gives representative activation contexts for these units. Four respond to numbers, quantities, or measurements, while the other two respond to punctuation or species names. The dominant pattern is therefore quantitative, with two units reflecting the surrounding syntactic or lexical context.

Table 10: Representative task-independent top-activation contexts for the six bottleneck units rendered in Figure 7. Bold marks the token or phrase most relevant to each tentative semantic hypothesis.

Bottleneck unit	Semantic hypothesis	Top-activating excerpts
L6 c380	measure/unit	"carapace length of 80–85 **millimetres**"; "usually 23–38 **cm**"; "a carapace length of 15 **mm**".
L6 c723	punctuation/function	Repeated comma-token activations in clauses such as "higher than that of H. americanus**,** and the European species" and around numeric punctuation.
L8 c264	quantity	"estimated that only **1** larva"; "not normally deeper than **50 m** (160 ft)"; "published in **1758**".
L8 c511	measure/place	"**48 kilometres or 30 miles northwest** of Gothenburg"; "for up to **12 months**"; "body length up to **60 centimetres**".
L10 c757	numeric/year	"published in **1758**"; "Homarus Weber, **1795**"; "Fabricius, **1775**".
L10 c627	species/name	Repeated activations on species-name fragments in "H. **americanus**", "H. **gammarus**", and "**Homarus** americanus".

Overall, the semantic audit gives the GreaterThan circuit a more concrete interpretation. Most displayed bottleneck units respond to numbers, years, quantities, or measurements in task-independent text, consistent with their high GreaterThan attribution. This suggests that attribution-selected SWD bottleneck units can carry recognizable semantic content. The audit is nonetheless limited: it covers six units with model-generated labels on the all-layer `mlp.c_proj` surface, whereas the held-out

sufficiency and necessity results are measured on the layer-8 single-matrix surface, so it complements rather than extends those causal results.

K TARGETED EDITING OF A SINGLE SWD BOTTLENECK UNIT

The main circuit experiments establish that SWD bottleneck units support targeted ablation. Here we ask a complementary question: can the read direction of one unit support a localized directional edit? Our design adapts the model-editing template of making a localized low-rank weight update and evaluating both target efficacy and locality on unrelated inputs (Meng et al., 2022). To remove replacement-fidelity differences from the comparison, every method starts from the same original dense GPT-2 Small layer-8 `mlp.c_proj` weight $\boldsymbol{W}$. The SWD intervention uses the read direction of one bottleneck unit to materialize the dense rank-one update induced by changing that unit's write direction. We include a random-unit control, a target-conditioned dense rank-one oracle, and rank-4 LoRA (Hu et al., 2022). The source SWD factorization is fitted separately for this study and uses $s = 0.75$, 768 bottleneck units, seed 0, and 131,072 WikiText-2 calibration tokens (Merity et al., 2017); it is therefore not the CE-matched $s = 0.5$ FineWeb-Edu factorization used in the circuit comparisons.

We study the prompt

`The opposite of up is` answer: `down`, foil: `left`.

The dense model assigns the answer an answer–foil logit margin of 3.261101. Let $\boldsymbol{a}_c = \boldsymbol{A}_{:,c} \in \mathbb{R}^{3072}$ be the read vector of bottleneck unit c, let $\boldsymbol{h}^\star \in \mathbb{R}^{1\times 3072}$ be the final-position input to `mlp.c_proj`, and let

$$\boldsymbol{d} = \boldsymbol{u}_{\text{ans}} - \boldsymbol{u}_{\text{foil}}, \qquad \overline{\boldsymbol{d}} = \frac{\boldsymbol{d}}{\|\boldsymbol{d}\|_2},$$

where $\boldsymbol{u}_{\text{ans}}, \boldsymbol{u}_{\text{foil}} \in \mathbb{R}^{768}$ are the column-vector forms of the corresponding unembedding rows. For requested local answer-direction shift δ, the SWD-guided edit is

$$\boldsymbol{W}_c(\delta) = \boldsymbol{W} + \alpha_c(\delta)\boldsymbol{a}_c\overline{\boldsymbol{d}}^\top, \qquad \alpha_c(\delta) = \frac{\delta}{(\boldsymbol{h}^\star\boldsymbol{a}_c)\|\boldsymbol{d}\|_2}.$$

It therefore satisfies

$$\boldsymbol{h}^\star\big(\boldsymbol{W}_c(\delta) - \boldsymbol{W}\big)\boldsymbol{d} = \delta.$$

This is the materialized dense increment induced by the corresponding write-row update, since

$$\boldsymbol{B}'_{c,:} = \boldsymbol{B}_{c,:} + \alpha_c(\delta)\overline{\boldsymbol{d}}^\top \quad \Longrightarrow \quad \boldsymbol{AB}' - \boldsymbol{AB} = \alpha_c(\delta)\boldsymbol{A}_{:,c}\overline{\boldsymbol{d}}^\top.$$

We apply this increment to the original dense $\boldsymbol{W}$, rather than using $\boldsymbol{AB}$ as the base, so that all editing methods are evaluated from the same model. The requested-shift grid is

$$\delta \in \{-2, -1, -0.5, 0, 0.5, 1, 2, 3\}.$$

Unit c205 was selected by screening eight candidate prompt–answer–foil triples for positive target activation and answer-direction write alignment, retaining the top 24 positive units per prompt and incorporating stored task importance and rank in the selection score. The *random-unit* control c540 was sampled with seed 17 from units whose absolute target activation lies between the 35th and 85th percentiles. It uses the same answer direction and requested-shift calibration as c205. The *target-conditioned dense rank-one oracle* replaces the SWD read vector by the normalized target activation,

$$\boldsymbol{r}_{\text{dense}} = \frac{\boldsymbol{h}^{\star\top}}{\|\boldsymbol{h}^\star\|_2}, \qquad \boldsymbol{W}_{\text{dense}}(\delta) = \boldsymbol{W} + \alpha_{\text{dense}}(\delta)\boldsymbol{r}_{\text{dense}}\overline{\boldsymbol{d}}^\top,$$

with $\alpha_{\text{dense}}(\delta)$ calibrated by the same local answer-direction constraint. Rank-4 LoRA uses $\Delta\boldsymbol{W} = \boldsymbol{L}_A\boldsymbol{L}_B$, with $\boldsymbol{L}_A \in \mathbb{R}^{3072\times 4}$ initialized as $0.01\mathcal{N}(0,1)$ and $\boldsymbol{L}_B \in \mathbb{R}^{4\times 768}$ initialized to zero. A separate LoRA update is trained for each requested final-margin shift using AdamW for 180 steps with learning rate 0.08, zero weight decay, and an L_2 coefficient of 10^{-4}. The Pareto comparison uses the realized final-margin change of every method.

At requested local shift $\delta = 3$, the c205 coefficient is $\alpha_{205} = 0.423220$, and the realized final answer–foil margin change is 0.215797. Mean dense-to-edited final-token KL over the seven non-target prompts is 4.02375×10^{-5}. On the separate 504-token global evaluation sample, the corresponding mean KL is 1.96415×10^{-5}. The realized margin change is much smaller than δ because

δ is defined locally, as the shift of the layer-8 `mlp.c_proj` output along the answer–foil unembedding direction; before reaching the logits, this shift passes through layers 9–11 and the final LayerNorm, which rescales the residual stream.

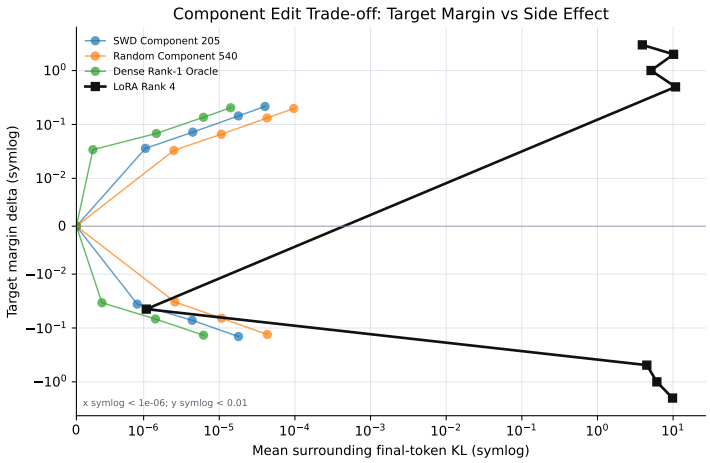

Figure 30: Target-margin change versus measured side effect for single-target edits applied to the same original dense weight matrix. The horizontal axis is mean final-token KL on seven unrelated factual prompts; the vertical axis is the change in the `down` versus `left` margin. Positive and negative edit settings form the upper and lower branches. At comparable margin change, farther left indicates lower measured side effect. The SWD curve uses the read direction of bottleneck unit c205; the dense rank-one oracle uses the target activation directly.

Figure 30 shows the expected ordering. The target-conditioned dense rank-one oracle gives the strongest tradeoff. The c205 edit is next: it opens farther vertically and remains farther left than the random-unit control. Rank-4 LoRA attains larger absolute margin changes, but at substantially higher non-target KL. Thus, on this audited target and probe set, an SWD-guided bottleneck-unit edit provides a more precise intervention than the random-unit and LoRA controls, though it does not match the target-conditioned dense rank-one oracle.

L ATTENTION BOTTLENECK UNITS: FROM STATIC QK GEOMETRY TO PROMPT-LOCAL EFFECTS

The workflow here combines the QK-circuit decomposition of attention (Elhage et al., 2021), sparse weight-space attention decomposition for circuit tracing (Franco & Crovella, 2024), and feature-level QK attribution (Kamath et al., 2025). Because a pre-softmax attention score is bilinear in its query- and key-side inputs, an additive feature decomposition rewrites that score as a sum of query–key feature-pair terms; prior work interprets those terms on concrete prompts and validates them with causal interventions. SWD supplies parameter-side query and key bottleneck units for the same analysis. We additionally begin with a reconstruction check so that any unit-level interpretation is tied to an attention pattern that remains close to the dense head.

The full-model experiment in Section 3.4 replaces every attention and MLP weight matrix across the transformer blocks. We use those factors to test whether sparse Q and K bottleneck units expose attention computations that are active on real tokens and sensitive to intervention. Following the motivation above, the analysis proceeds from reconstruction fidelity, to a weight-space candidate screen, to prompt-level replay and ablation.

L.1 ATTENTION FACTORIZATION FIDELITY

We first measure activation-weighted relative reconstruction SSE for the packed Q/K/V weight matrix `attn.c_attn` and the attention output weight matrix `attn.c_proj` in all 12 GPT-2 Small layers. As shown in Figure 31 (top), the Q/K/V reconstruction error remains below 4% in every layer. The attention output matrix is less accurate in the middle layers, where its error reaches approximately 6%. The prompt-level analysis below therefore uses the better-preserved `attn.c_attn` factors and separately checks the reconstructed attention pattern against the dense head.

L.2 STATIC QK CANDIDATE SCREEN

For one head h, write the reconstructed query and key at token positions t and u as

$$\boldsymbol{q}_t^h = \sum_a z_a(\boldsymbol{h}_t)\boldsymbol{b}_{a,h}^{\mathrm{Q}},$$
$$\boldsymbol{k}_u^h = \sum_b z_b(\boldsymbol{h}_u)\boldsymbol{b}_{b,h}^{\mathrm{K}}.$$

Expanding their scaled dot product separates each attention score into bottleneck-unit-pair terms:

$$\mathrm{score}_{tu}^h = \sum_{a,b} z_a(\boldsymbol{h}_t)z_b(\boldsymbol{h}_u)\gamma_{ab}^h,$$
$$\gamma_{ab}^h = \frac{\langle \boldsymbol{b}_{a,h}^{\mathrm{Q}}, \boldsymbol{b}_{b,h}^{\mathrm{K}} \rangle}{\sqrt{64}}.$$

The static coefficient γ_{ab}^h measures the interaction between a Q write direction and a K write direction; the activation product $z_a(\boldsymbol{h}_t)z_b(\boldsymbol{h}_u)$ determines whether that interaction contributes on a particular prompt and token pair.

For each of 12 layers and 12 heads, we retain the 16 Q and 16 K bottleneck units with the largest head-specific write norms, evaluate all 256 cross-pairs, and summarize the head by its largest $|\gamma_{ab}^h|$. Figure 31 (bottom) shows that the strongest candidates occur mainly in later layers. We advance the five strongest pairs in every head to prompt replay.

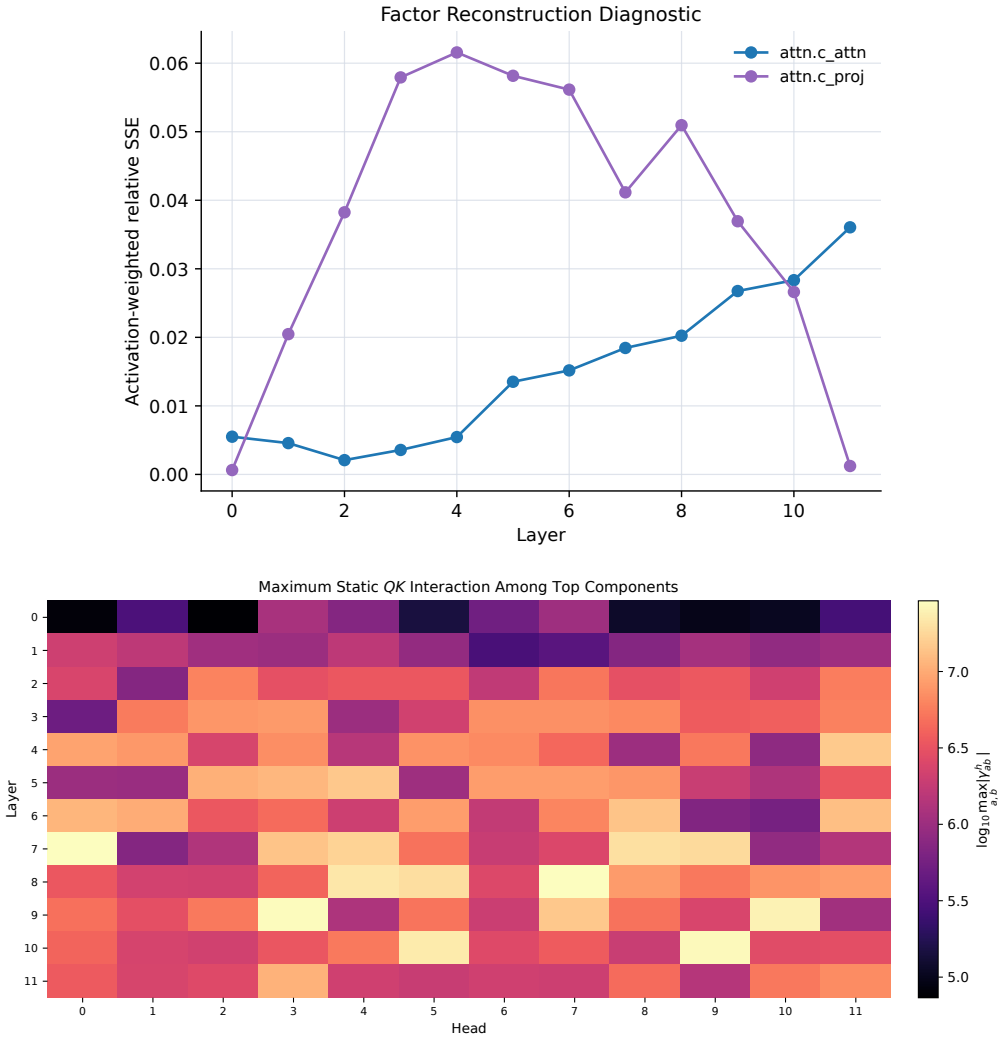

Figure 31: Attention reconstruction and static QK candidate screen. Top: activation-weighted relative reconstruction SSE for the packed Q/K/V and attention output matrices across GPT-2 Small. Bottom: the largest absolute static QK coefficient among the retained 16×16 bottleneck-unit pairs in each layer and head.

L.3 Prompt Replay and Bottleneck-Unit Intervention

We evaluate the five strongest static pairs from each layer and head on three prompts, for

$$12 \text{ layers} \times 12 \text{ heads} \times 5 \text{ pairs} \times 3 \text{ prompts} = 2{,}160$$

prompt–head–pair evaluations. The clearest response occurs on `When Mary gave John the book, he thanked` in layer 9, head 3, for pair q266×k64.

Before interpreting the pair, we compare the dense attention pattern with the pattern obtained after replacing the packed Q/K/V weight matrix by its SWD reconstruction. Figure 32 (top) shows that the main attention structure is retained; the mean dense-to-reconstructed attention KL is 0.0405. We then evaluate the full pair contribution

$$z_{266}(\boldsymbol{h}_t) z_{64}(\boldsymbol{h}_u) \gamma^h_{266,64}$$

at every causally valid query–key position. As shown in Figure 33, the pair contributes negatively toward the first key, `When`, and positively toward most later keys, with signed mean $+4.26$ over valid positions.

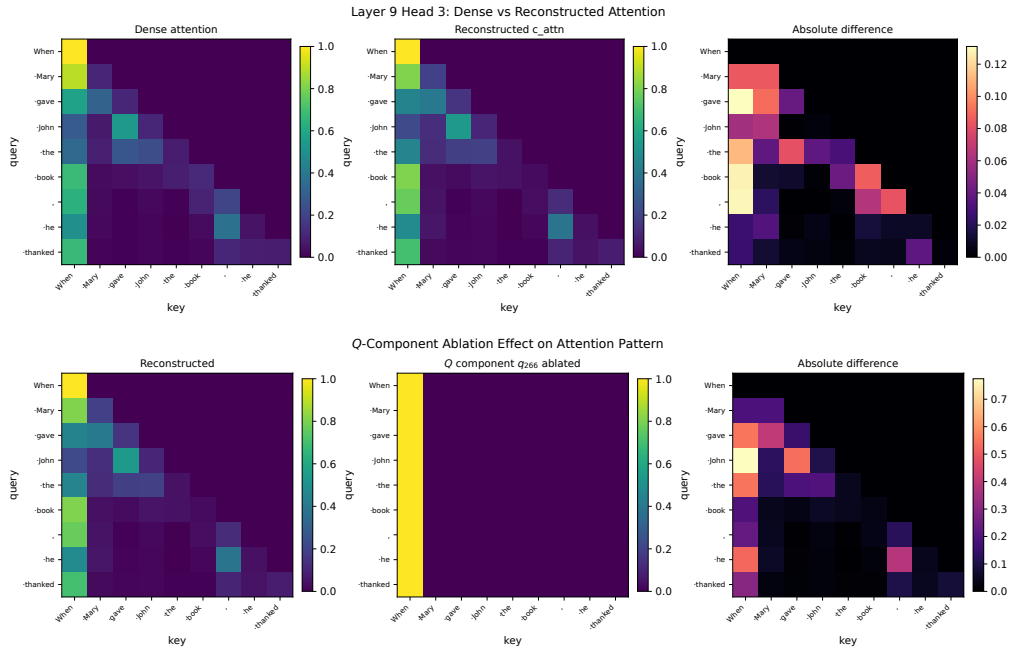

Figure 32: Prompt-level validation and intervention for layer 9, head 3. Top: dense attention, SWD-reconstructed attention, and their absolute difference; mean dense-to-reconstructed attention KL is 0.0405. Bottom: SWD-reconstructed attention before and after ablating query bottleneck unit q266, followed by their elementwise absolute difference. Relative to the intact reconstruction, ablating q266 produces a mean attention KL of 2.496 and a maximum absolute change of 0.775 in any attention probability.

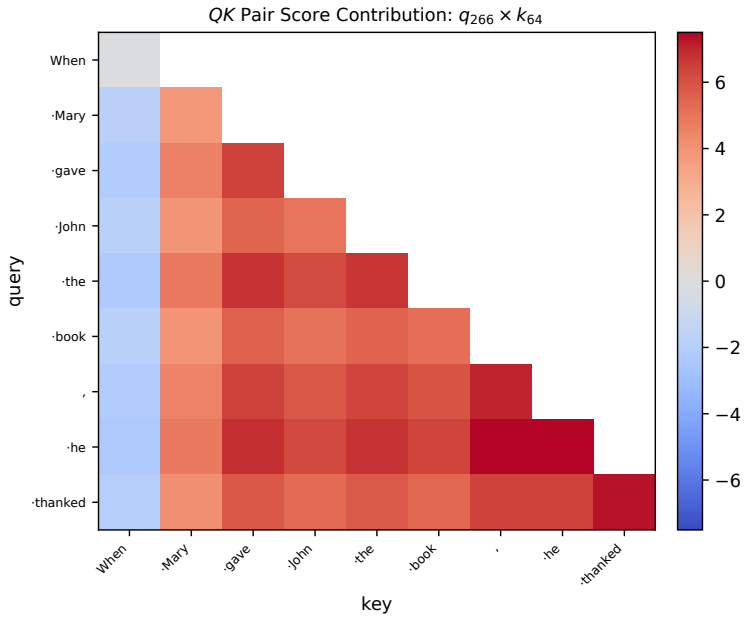

Figure 33: Token-level pre-softmax score contribution of q266×k64 in layer 9, head 3. Each cell shows $z_{266}(\boldsymbol{h}_t) z_{64}(\boldsymbol{h}_u) \gamma^h_{266,64}$ for one causally valid query–key pair; the signed mean is $+4.26$.

Finally, we remove q266 from the reconstructed Q slice and recompute the head. Attention shifts sharply toward the first token: relative to the intact reconstructed pattern, mean attention KL rises to 2.496 and the maximum probability change is 0.775 (Figure 32, bottom). This reversal agrees with the signed contribution map: removing q266 eliminates the unit's suppression of `When` and its support for later keys.

Together, these checks close the chain from static geometry to causal effect. SWD closely reconstructs the selected head, static QK geometry identifies q266×k64, prompt replay shows a coherent token-level contribution, and ablating q266 reverses that pattern. This is a single selected attention case, but it shows that SWD's attention bottleneck units can expose a concrete and manipulable part of a head's computation.

M Interaction Patterns among MLP Bottleneck Units

Feature-based circuit analyses express an MLP output as a sum of component contributions (Dunefsky et al., 2024), while work on superposition shows that the corresponding features can overlap (Elhage et al., 2022). More generally, feature interactions can be measured as departures from additivity introduced by nonlinear activations (Tsang et al., 2018). We therefore examine two complementary phenomena: correlation between additive output contributions after GELU, and contextual non-additivity when bottleneck-unit contributions are combined before GELU.

We analyze the two MLP weight matrices separately in GPT-2 Small layer 8. For the output matrix `mlp.c_proj`, bottleneck-unit contributions are combined linearly after GELU, so we measure correlation between their residual-stream contributions. For the input matrix `mlp.c_fc`, the vector contributions of the bottleneck units are summed before GELU, so we measure their pairwise contextual finite difference across the nonlinearity.

M.1 c_proj: Linear Contributions Are Usually Weakly Correlated

The linear analysis uses all 768 bottleneck units of the layer-8 `mlp.c_proj` factorization with $s = 0.75$ and 2,048 held-out token activations. Bottleneck unit i reads a scalar $z_i = \boldsymbol{h}\boldsymbol{A}_{:,i}$ and contributes the vector $z_i \boldsymbol{B}_{i,:}$ to the residual stream. We define its output-variance energy as

$$\mathrm{Var}(z_i)\|\boldsymbol{B}_{i,:}\|_2^2.$$

The 96 highest-energy bottleneck units are retained for the readable matrix views in Figure 34; distributional statistics use all 768 units.

For each bottleneck-unit pair, we measure absolute token-level read correlation, absolute cosine similarity between write directions, and the correlation of the complete residual contributions. Read and write overlap are each common in isolation, but they rarely occur for the same pair. Across all units, the 95th-percentile absolute read correlation is 0.207 and the 95th-percentile absolute write cosine is 0.071, whereas the 95th-percentile absolute contribution correlation is only 0.0049; even its 99th percentile is 0.0089.

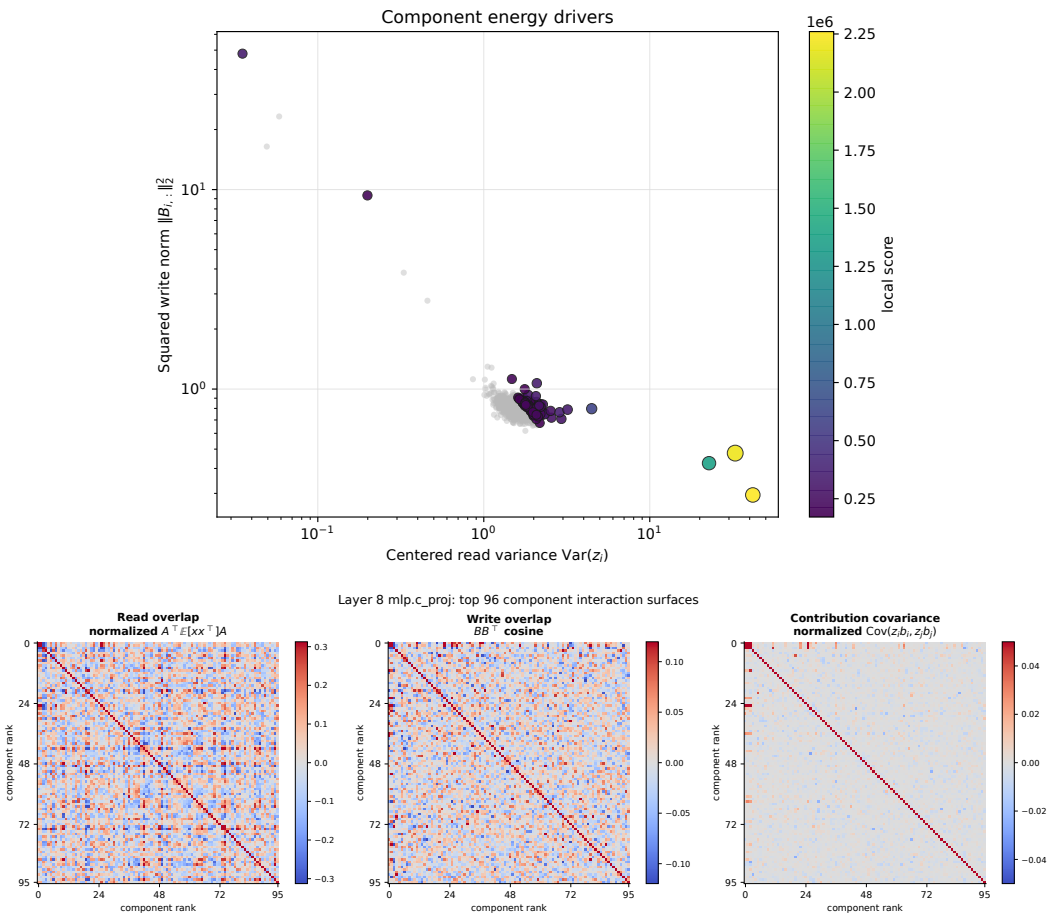

Figure 34: Selection and overlap structure for layer-8 `mlp.c_proj`. Top: read variance versus squared write norm; the 96 bottleneck units with highest output-variance energy are highlighted. Bottom: read correlation, write-direction cosine, and complete contribution correlation for those units. Read and write structure is visible separately but largely disappears when combined into full residual contributions.

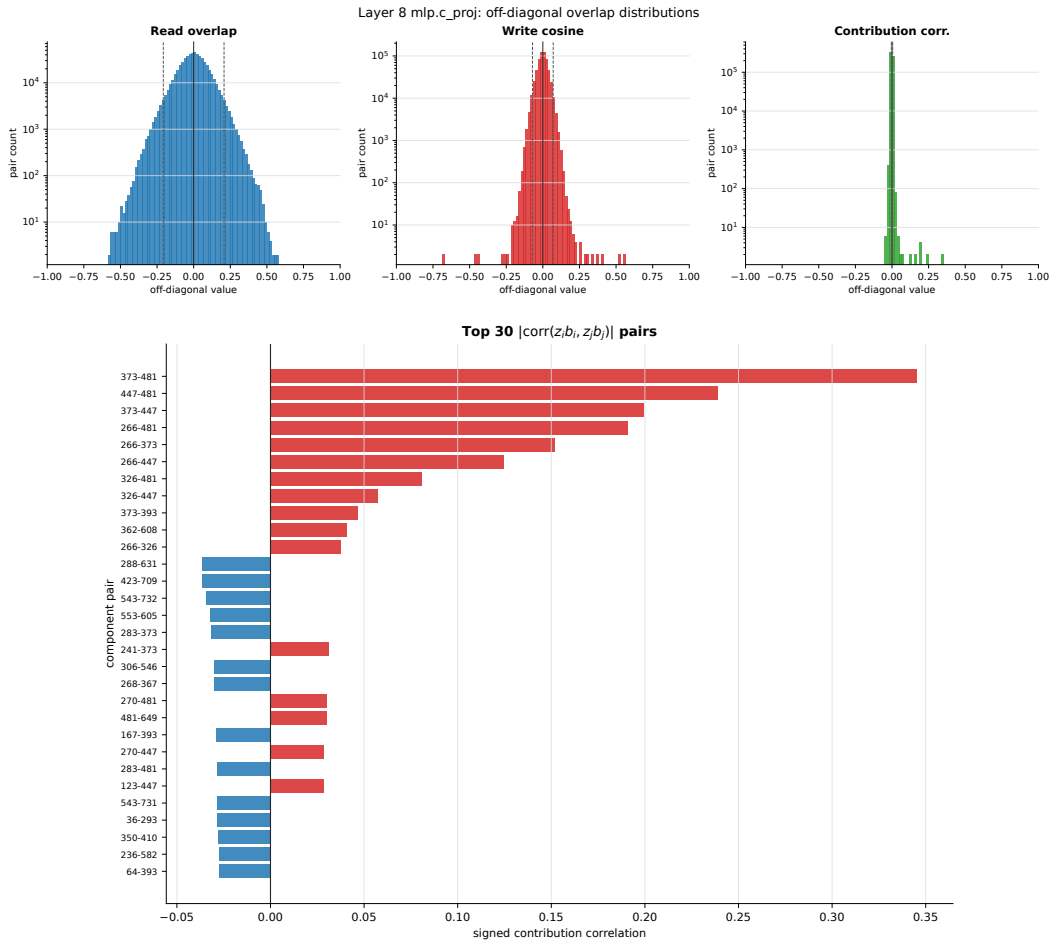

Figure 35: Linear bottleneck-unit dependence in layer-8 `mlp.c_proj`. Top: off-diagonal pair distributions over all 768 units. Bottom: the 30 pairs with largest absolute contribution correlation. Most contribution correlations concentrate near zero, while a small tail is organized around recurring units; c373–c481 reaches 0.345.

The exceptional tail is compact. Bottleneck units 373, 447, 481, and 266 recur among the strongest positive pairs; the leading c373–c481 pair has contribution correlation 0.345. In that pair, opposed read patterns combine with opposed write directions, so the two sign reversals yield positively correlated residual contributions. Thus, linear `c_proj` bottleneck units are usually weakly correlated as complete pathways, with a small set of structured exceptions.

M.2 `c_fc` AND GELU: CONTEXTUAL PAIR INTERACTIONS

The nonlinear diagnostic uses a separately fitted, overcomplete factorization of `mlp.c_fc` in all 12 layers. Each layer uses inner width $m = 1{,}536 = 2\min(d_{\text{in}}, d_{\text{out}})$, so $\boldsymbol{A} \in \mathbb{R}^{768\times1536}$ and $\boldsymbol{B} \in \mathbb{R}^{1536\times3072}$, and sparsity $s = 0.875$. The layers are factorized sequentially on the same 4.19M FineWeb-Edu calibration tokens (4,096 sequences of 1,024 tokens): the calibration activations of each layer are recomputed with the replacements of all lower layers installed, and the solver's penalty split between the two factors is selected per layer. We analyze layer 8 on 8,192 held-out WikiText-2 validation tokens, replayed with the layer 0–7 `c_fc` replacements installed, so that the layer-8 inputs follow the sequentially reconstructed model. To obtain a readable pair matrix, we retain 128 units with the largest factor-rescaling-invariant pre-GELU downstream energy,

$$R_i = \mathbb{E}_t[z_{t,i}^2]\,\|\boldsymbol{B}_{i,:}\boldsymbol{W}_{\text{proj}}\|_2^2,$$

giving $128 \times 127/2 = 8{,}128$ unordered pairs.

Let $\boldsymbol{c}_{t,i} = z_{t,i}\boldsymbol{B}_{i,:}$ be unit i's vector contribution to the c_fc output, and let

$$\boldsymbol{h}_t = \boldsymbol{b}_{\text{fc}} + \sum_a \boldsymbol{c}_{t,a}$$

be the complete reconstructed preactivation. For each pair, we compute the contextual finite difference

$$\begin{aligned}\boldsymbol{\Delta}_{t,ij} = \ & \text{GELU}(\boldsymbol{h}_t) - \text{GELU}(\boldsymbol{h}_t - \boldsymbol{c}_{t,i}) - \text{GELU}(\boldsymbol{h}_t - \boldsymbol{c}_{t,j}) \\ & + \text{GELU}(\boldsymbol{h}_t - \boldsymbol{c}_{t,i} - \boldsymbol{c}_{t,j}).\end{aligned}$$

We then project this interaction through the original MLP output matrix, $\boldsymbol{q}_{t,ij} = \boldsymbol{\Delta}_{t,ij}\boldsymbol{W}_{\text{proj}}$. Define the contextual marginal residual effect of unit i as

$$\boldsymbol{m}_{t,i} = \left[\text{GELU}(\boldsymbol{h}_t) - \text{GELU}(\boldsymbol{h}_t - \boldsymbol{c}_{t,i})\right]\boldsymbol{W}_{\text{proj}}, \qquad E_i = \sum_t \|\boldsymbol{m}_{t,i}\|_2^2.$$

The normalized interaction magnitude is

$$R_{ij} = \frac{\left(\sum_t \|\boldsymbol{q}_{t,ij}\|_2^2\right)^{1/2}}{(E_i E_j)^{1/4}}.$$

Pair interactions are widespread but typically moderate. Across the 8,128 pairs, mean R is 0.093, the median is 0.089, the 95th percentile is 0.132, the 99th percentile is 0.154, and the maximum is 0.221. In total, 2,130 pairs have $R \geq 0.10$, 113 have $R \geq 0.15$, and four have $R \geq 0.20$. Figure 36 shows a dense central band together with a thinner upper tail. The brighter first rows and columns indicate that several high-energy units interact moderately with many partners, while isolated bright cells reveal additional pair-specific effects.

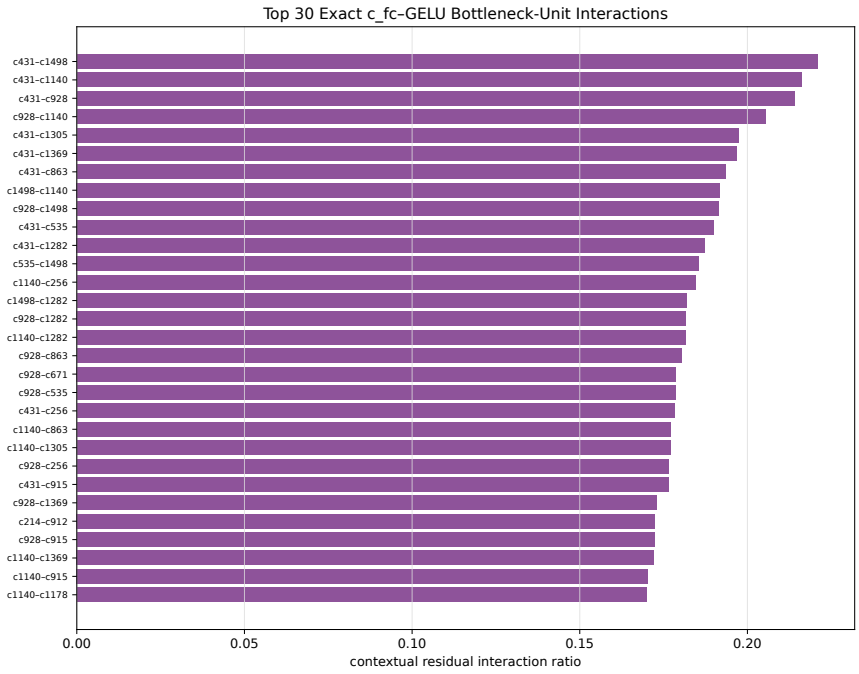

Figure 37: The 30 strongest contextual GELU-mediated pairs, ranked by residual-stream interaction ratio. Units c431, c928, c1140, and c1498 recur throughout the upper tail. The leading c431–c1498 pair reaches $R = 0.221$; its interaction residual RMS is 2.096, compared with marginal residual RMS values 11.127 and 8.089 for the two units.

The two MLP locations therefore show complementary structure. After GELU, most c_proj bottleneck paths make weakly correlated residual-stream contributions, with a compact set of exceptions. Across GELU, the c_fc finite differences reveal a dense field of moderate interactions and a thinner

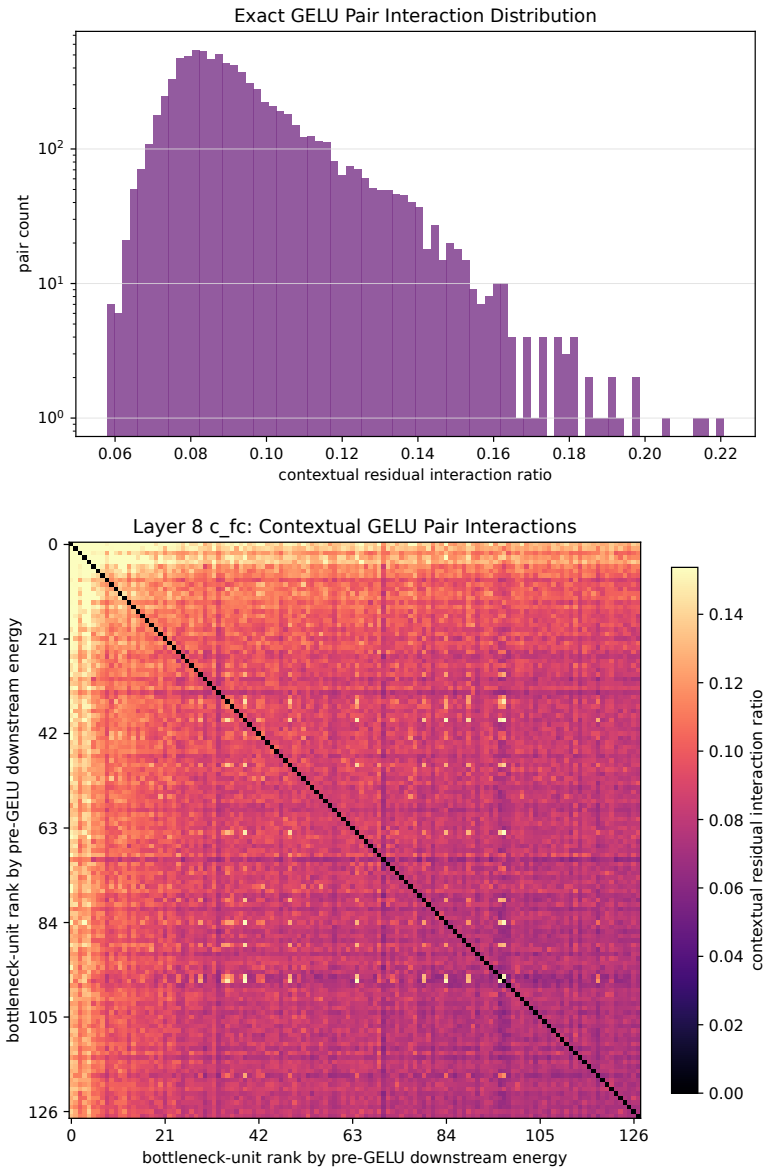

Figure 36: Contextual GELU-mediated interactions among 128 high-energy layer-8 `mlp.c_fc` bottleneck units. Top: distribution of residual-stream interaction ratios over all 8,128 unordered pairs. Bottom: the same ratios ordered by pre-GELU downstream energy. Interactions are broadly distributed, with stronger values concentrated around several recurring units and a smaller set of pair-specific cells.

tail organized around recurring high-energy units. Thus, output-projection paths are usually well described by their additive contributions, whereas the effects of some input-projection paths are more informative when examined together with their strongest interaction partners.