EXACT FUNCTIONAL SOLUTION SPACE: AUDITING NEURAL ANALYSES ACROSS EXACT-EQUIVALENT PARAMETERIZATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural-network analyses are typically computed from a particular parameterization but often interpreted as properties of the learned solution. This interpretation is problematic because different parameterizations can realize exactly the same input–output function while changing internal quantities and the outcomes of downstream procedures. We address this problem using **Exact Functional Solution Space (EFSS)**, which identifies a neural solution with its realized function rather than with a particular parameter vector. This yields a simple criterion: a solution-level claim should remain consistent across exact-equivalent parameterizations under the appropriate semantics of its output. We turn this criterion into an output-aware audit that distinguishes direct invariance, correspondence under relabeling or transport, and consistency of model-producing procedures. Across Transformer interventions, pruning, and model editing, we find that similar task performance can conceal representation-dependent scientific judgments, while unstable internal measurements can leave reported decisions unchanged. These findings show that consistency must be assessed at the level of the claim being made. EFSS makes exact functional equivalence a practical basis for tracing such dependence through an analysis and guiding targeted repairs under the tested transformations. By separating the consistency of scientific conclusions from task utility, the framework helps researchers assess what their analyses establish about a learned function.

1 INTRODUCTION

A trained neural network is often stored as a set of weights. Yet the function computed by those weights can be represented in many different ways. Different parameter values can therefore implement the same input–output behavior.

This matters whenever internal quantities are used as scientific evidence. Such evidence can guide a pruning allocation or a decision to accept a model edit. We may ask which hidden unit is important, which representations are similar, or which weights should be removed. Studies of indirect object identification (IOI) exemplify component-level interpretation (Wang et al., 2022). A change that preserves the function may only rename units, but it may also change an intervention or produce a different sparse model. We need to know *which conclusions belong to the function* and *which belong only to its representation.*

Prior work explains important parts of this non-uniqueness. Studies of realization-map fibers, parameter identifiability, and hidden symmetries characterize when different parameterizations represent the same function and how permutation or scaling changes arise (Grigsby et al., 2023; 2025; Grillo & Montúfar, 2026; Ramakrishnan, 2026; Zhao et al., 2025; Dong & Cheng, 2026). Implementation Invariance and representation-comparison methods ask when analysis results remain comparable as internal representations change (Sundararajan et al., 2017; Kornblith et al., 2019); related work studies reparameterization-aware interventions and equivariant models for particular transformations or architectures (Asiaee, 2026; Tran et al., 2025). Most closely, Causal Mechanism Reduction (CMR; Asiaee, 2026) tests positive-scaling invariance of reductions and their interventional fidelity. Building on these principles, we distinguish intermediate outputs, reported decisions,

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.

and utility. Our experiments reveal when representation dependence changes a decision, when another stage compensates it, and when a coarser conclusion stays intact. Pruning methods provide a direct setting in which an analysis constructs a new model, from magnitude and curvature criteria to activation-aware and one-shot methods (LeCun et al., 1990; Hassibi & Stork, 1993; Sun et al., 2024; Frantar & Alistarh, 2023). These works explain how equivalent representations arise and offer useful local comparison principles (extended comparison in Appendix A).

The question we address is **when two parameterizations compute the same function, what should count as the same scientific result?** EFSS supplies a common functional identity for this audit. If two parameterizations agree on every input, we treat them as the same solution.

We define the **E**xact **F**unctional **S**olution **S**pace (EFSS) by identifying all parameterizations that realize the same function on a declared input domain. Section 2 formalizes this identity and distinguishes complete functional fibers from known symmetry orbits.

The audit compares function-level scalars directly, labeled outputs after transport, and model-producing procedures through their resulting functions. These declared *Output Semantics* also specify the claim being tested: a selected unit set, a layer allocation, and an edit-acceptance verdict need not respond alike to a change of representation.

The pruning experiments make the distinction concrete: layer budgets change even when their reconstruction effects offset in final NLL. Model editing (Meng et al., 2022) provides a second decision: native fine-tuning changes acceptance in about 8% of paired comparisons despite unchanged rewrite success. Conversely, unstable unit masks preserve the selected top candidate layer. The audit thus identifies which claim changes and supports a specific diagnosis, rather than inferring decision instability from an intermediate discrepancy.

Our contributions are fourfold:

1. **An architecture-general framework.** EFSS organizes functional identity, certified symmetry families, and output-specific comparisons.
2. **A claim-level audit.** We compare intermediate outputs, reported decisions, and utility under declared transports.
3. **Decision-level evidence.** We establish pruning-allocation changes with compensating reconstruction effects and edit-acceptance changes despite stable rewrite success.
4. **Mechanism-guided diagnosis.** We separate selection, reconstruction, and coordinate-constraint effects, and assess consistency repair alongside task utility.

2 Exact Functional Neural Solutions

We identify neural solutions by their realized functions. Fibers and quotients organize this identity; certified permutations, scalings, and factorization freedoms lie inside fibers without necessarily exhausting them.

2.1 From Parameterizations to Exact Functional Identity

Fix a neural architecture A. Let Θ_A be its parameter space, X its declared full input domain, and let the output interface be fixed as part of the system specification. Each parameterization $\theta \in \Theta_A$ realizes an input–output function f_θ. The realization map is

$$R_A : \Theta_A \longrightarrow \mathcal{F}_A, \qquad R_A(\theta) = f_\theta, \tag{1}$$

where $\mathcal{F}_A = R_A(\Theta_A)$ is the set of functions realizable by A.

The parameterization and the function it realizes are different objects. For example, the two-layer scalar computation

$$\boxed{\theta :\ h = 2x, \quad y = 3h = 6x} \quad \overset{\text{same function}}{\longleftrightarrow} \quad \boxed{\theta' :\ h' = 4x, \quad y' = 1.5h' = 6x}$$

has different hidden values and downstream weights, but both parameterizations compute $y = 6x$. In neural networks, the same phenomenon appears through hidden-unit permutations, positive scalings, reciprocal gauges, and factorization freedoms.

The relevant identity is therefore *functional* rather than parametric. We use $\equiv$ throughout this paper to denote exact equality of realized functions:

$$\theta \equiv \eta \quad \Longleftrightarrow \quad f_\theta(x) = f_\eta(x) \quad \forall x \in X. \tag{2}$$

This equivalence relation defines the **same exact neural solution** under the declared specification (Appendix B.1).

The input domain and output interface matter: equality of logits, probabilities, and decoded decisions are different claims. Finite test agreement, numerical residuals, or parameter proximity cannot establish $\theta \equiv \eta$ on all of X; exactness requires an all-domain identity.

2.2 Functional Fibers and the Exact Solution Space

For $f \in \mathcal{F}_A$, its functional fiber is

$$M_f = R_A^{-1}(\{f\}) = \{\theta \in \Theta_A : f_\theta = f\}. \tag{3}$$

The fiber contains *every parameterization* that realizes the same function. Hence an exact neural solution is not one parameter point but the whole equivalence class of that point:

$$[\theta]_\equiv = M_{f_\theta}. \tag{4}$$

The fibers form a disjoint partition of Θ_A.

Identifying each fiber as one element gives the Exact Functional Solution Space,

$$\mathcal{S}_A^{\mathrm{ex}} = \Theta_A/\equiv, \qquad q_A(\theta) = [\theta]_\equiv. \tag{5}$$

The elements of $\mathcal{S}_A^{\mathrm{ex}}$ are exact functional solutions, not selected parameter vectors. The realization map descends to

$$\widetilde{R}_A : \mathcal{S}_A^{\mathrm{ex}} \longrightarrow \mathcal{F}_A, \qquad \widetilde{R}_A([\theta]_\equiv) = f_\theta. \tag{6}$$

Proposition 1 (Quotient–realization bijection). The map $\widetilde{R}_A$ is a well-defined bijection. This standard quotient construction follows directly from the fiber definition; the proof is in Appendix B.2. Thus parameter space contains representations, whereas EFSS contains exact functional solutions.

2.3 A Unified Language for Neural Solution Identity

Neural networks admit many function-preserving transformations, including hidden-unit permutations, positive-homogeneous scalings, reciprocal scalings, matrix-factorization redundancies, and gauge transformations. These phenomena have been studied using different architectural and mathematical descriptions (Grigsby et al., 2023; 2025; Grillo & Montúfar, 2026; Ramakrishnan, 2026; Zhao et al., 2025; Dong & Cheng, 2026; Tran et al., 2025).

EFSS treats these transformations as special motions inside one exact functional solution, rather than as separate definitions of solution identity. Let a group G act on Θ_A and suppose

$$R_A(g \cdot \theta) = R_A(\theta) \qquad \forall g \in G. \tag{7}$$

Then its ***orbit*** satisfies

$$G \cdot \theta \subseteq M_{f_\theta}. \tag{8}$$

A permutation orbit, scaling orbit, or gauge orbit is therefore a certified representation family within one exact solution; it does not itself define exact solution identity.

This observation gives a common form to several familiar questions. For a known symmetry group G, identifiability asks whether

$$G \cdot \theta = M_{f_\theta} \tag{9}$$

holds. If it does, the known orbit describes the complete fiber for that function. If it does not, there are exact-equivalent representations that the known symmetry cannot reach. ***Canonicalization*** asks whether one can select a representative consistently from a fiber. Detailed architecture-specific examples and transformation certificates are given in Appendix B.3.

Symmetry describes certified motions within a fiber; identifiability asks whether they exhaust it; redundancy measures representation multiplicity; canonicalization selects a representative. These are distinct questions about the same functional identity.

2.4 Beyond Known Symmetries: Residual Exact Structure

The inclusion in Eq. (8) need not be an equality. After quotienting by a known symmetry group, the residual structure can be written as

$$H_f^G = M_f/G. \tag{10}$$

If H_f^G is a singleton, the selected group accounts for the representation multiplicity of that fiber. Otherwise, discrete branches, continuous residual families, or singular structures remain after the known symmetry has been removed. A concrete residual-fiber witness, including a singular-fiber example, is given in Appendix B.4.

The basic hierarchy is therefore

$$\Theta_A \longrightarrow \Theta_A/G \longrightarrow \mathcal{S}_A^{\mathrm{ex}} \cong \mathcal{F}_A. \tag{11}$$

The intermediate quotient identifies known symmetry orbits; EFSS identifies complete functional fibers. The last identification therefore follows from **function equality** without assuming completeness of a chosen symmetry group.

This establishes what counts as one solution and what kinds of representations may lie inside it. The next section asks which quantities computed from a representative can legitimately be called properties of that solution.

3 Solution-Level Claims and Exact-Functional Audit

We give an architecture-general criterion for deciding whether a claim computed from a parameterization is well-defined for the realized function. Transformers offer certified transformations and practical pruning pipelines to trace representation dependence to downstream behavior.

3.1 When Does a Quantity Belong to the Solution?

Let

$$H : \Theta_A \longrightarrow Z \tag{12}$$

be a reported quantity. Depending on the analysis, $H(\theta)$ may be a sharpness value, importance score, activation statistic, ranking, pruning score, or model descriptor.

For a quantity H interpreted as a property of the realized function, directly comparable outputs must satisfy

$$\theta \equiv \eta \quad \Longrightarrow \quad H(\theta) = H(\eta). \tag{13}$$

This is not merely a preference for robustness. It is the condition under which H is well-defined on the quotient solution space.

Proposition 2 (Solution-level factorization). The quantity H is constant on every exact functional fiber if and only if there is a unique map $\overline{H} : \mathcal{S}_A^{\mathrm{ex}} \to Z$ satisfying

$$H = \overline{H} \circ q_A. \tag{14}$$

Here $\circ$ denotes function composition: $(\overline{H} \circ q_A)(\theta) = \overline{H}(q_A(\theta))$. The proof is given in Appendix C.1.

If $\theta \equiv \eta$ but $H(\theta) \neq H(\eta)$, the measurement is representation-dependent. It may remain useful for a chosen parameterization; the audit tests its interpretation as a property of the realized function.

3.2 Invariance, Equivariance, and Output Semantics

Requiring every output to satisfy raw equality is *too strong* for labeled or coordinate-dependent objects. If hidden units are permuted, a neuron-wise score vector may change position while the score associated with each underlying unit remains the same.

For a scalar or coordinate-independent descriptor claimed to characterize the realized function, the natural relation is invariance:

$$H(g \cdot \theta) = H(\theta). \tag{15}$$

For a labeled or structured output, the transformation induces a correspondence τ_g, and the appropriate relation is equivariance:

$$H(g \cdot \theta) = \tau_g(H(\theta)) \,. \tag{16}$$

The raw arrays need not be equal; they should agree after the declared *relabeling* or *transport*. Transport is predeclared and respects identity, composition, and stabilizers (Appendix C.5). Masks, subspaces, and bases are compared in the same way after being placed in a common frame.

When an analysis or procedure constructs a new parameterized model, output consistency has another level. Let $\mathcal{P}$ return a model and let $\widehat{g}$ transport that model from the original coordinates to the transformed coordinates. We compare

$$\mathcal{P}(g \cdot \theta) \quad \text{with} \quad \widehat{g}(\mathcal{P}(\theta)) \,, \tag{17}$$

and, when the output is a model, compare the functions they realize. This avoids treating an untransported raw mask as the scientific result.

We call this declared comparison rule the *Output Semantics* of an analysis: scalars with a function-level interpretation are compared by equality, labeled or structured outputs after the appropriate relabeling or transport, and model-producing procedures at the resulting-function level. Raw disagreement alone is therefore not a solution-level failure; the relevant question is whether the outputs disagree under their declared semantics.

3.3 Exact-Functional Well-Definedness Audit

Construct representatives $\theta, g_1 \cdot \theta, g_2 \cdot \theta, \ldots$ using transformations certified before the analysis. Each must satisfy

$$g_i \cdot \theta \equiv \theta. \tag{18}$$

Analytic identities provide the exactness certificate; numerical residuals check the implementation and finite-precision behavior, but do not replace the all-domain identity.

Before applying H, we declare its Output Semantics and the corresponding comparison relation. We then run the same analysis independently on each exact-equivalent representative and test whether the prescribed relation is preserved.

The audit has an important asymmetry. A single certified pair that violates the required relation is enough to refute the universal claim that H is a solution-level property over all representations. Conversely, agreement over several transformations does not prove agreement over the complete fiber, because a known orbit may be a strict subset of that fiber:

$$G \cdot \theta \subsetneq M_{f_\theta}. \tag{19}$$

An empirical pass therefore means consistency on the tested exact transformation family, under the declared Output Semantics. It does not certify every exact-equivalent parameterization. A step-by-step justification of this asymmetry is given in Appendix C.3.

The report must declare the claim being tested. If an intermediate output H is summarized by a decision d, the relevant decision is $d \circ H$; changes in H need not change that decision. Agreement on a scalar quality metric likewise need not identify the same structured output. We therefore compare intermediate results, reported decisions, and utility separately (Appendix G.1). Procedure hyperparameters and constraints are part of the specification; coordinate dependence can be a legitimate design choice, requiring a representation-qualified claim rather than a function-level interpretation.

3.4 Consistency Repair

We use *consistency repair* to mean modifying an analysis or procedure so that exact-equivalent representations better satisfy the comparison relation declared for that output. Its goal is not merely to make two numbers close, but to recover ***the same scientific object*** across the tested representatives.

One repair strategy constructs an *orbit-consistent* observable. If two representation-local quantities transform by compensating factors, their appropriate composition can cancel the representation dependence. The resulting observable is consistent under the declared transformation family; full-fiber validity requires a separate argument.

A second strategy aligns representatives before applying a coordinate-dependent procedure. For a deterministic normalization or balanced representative $C(\theta)$ defined for a declared transformation family (see Appendix B.3 for the transformation families used in this paper), the construction is

$$\theta \longrightarrow C(\theta) \longrightarrow \mathcal{P}(C(\theta)) \longrightarrow T_\theta(\mathcal{P}(C(\theta)))\,, \tag{20}$$

where T_θ transports the procedure output back to the original coordinates. The first strategy changes the observable; the second changes the frame in which the procedure is applied.

Neither strategy is a universal canonicalizer. A normalization defined for one architecture or gauge family need not select a representative continuously over the full exact quotient. Likewise, improved representation consistency does not imply improved task utility. These are separate questions and are evaluated separately in the experiments. Detailed derivations of the repair are given in Appendix C.4.

4 Experiments: Auditing Analyses on the Same Exact Solution

We connect exact functional identity to intermediate outputs, pruning decisions, and downstream behavior. The audit follows

$$\theta \equiv g\theta \;\longrightarrow\; \text{analysis or procedure} \;\longrightarrow\; \text{declared comparison} \;\longrightarrow\; \text{verdict}.$$

4.1 Experimental protocol

Local studies use GPT-2 (Radford et al., 2019) and Qwen2.5-0.5B/1.5B (Qwen Team, 2024), with a held-out 7B extension (Appendix D.9). GPT-2 channel permutations and Qwen positive diagonal SwiGLU gauges analytically preserve logits; numerical probes check implementation, with uncompensated scaling as a function-changing control (Appendix D.1). Local procedure/task tests prune one layer-0 down projection; allocation covers all MLP down projections in Qwen2.5-1.5B and TinyLlama-1.1B-Chat (Zhang et al., 2024).

The allocation and editing studies use a base and four fixed dyadic gauges per checkpoint, $d_i \in \{1/2, 1, 2\}$, in FP32 with TF32 disabled. Protocols were locally frozen before confirmation. Paired-cluster inference includes multiple-comparison and three-application corrections; gauges are not independent models. Execution details, controls, and sampling units are distinguished in Appendices D.10 and G.

4.2 Transport distinguishes representational from scientific disagreement

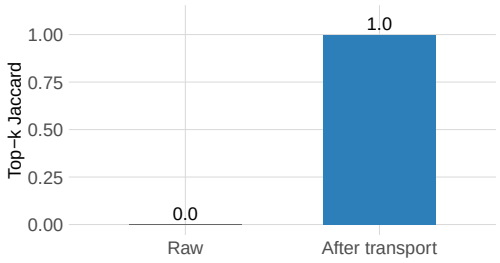
In the GPT-2 positive control, a hidden-channel permutation preserves the dense computation while changing channel labels: raw ranking overlap is $J_{\mathrm{raw}} = 0$, but inverse transport gives $J_{\mathrm{transported}} = 1$. Figure 1(a) thus recovers the same scientific object after relabeling.

Selector to intervention. In the Qwen reciprocal gauge, the dense function is preserved while an activation-only selector changes. The raw selector Jaccard is 0.2427 in the reported worst case, whereas the composed selector recovers 1.0. The direct intervention experiment separates selector change from intervention change (Figure 1(b)): the fixed-mask control has median gap 7.1×10^{-7}, while independently selected activation and downstream-weight masks have median gaps 0.176 and 0.530. The composed selector has zero gap. The full factorial grid is reported in Appendix D.3. A held-out check on indirect object identification (IOI), a recipient-name prediction task (Wang et al., 2022), also finds activation-induced layer-order reversals, with consistent composed/Taylor controls (Appendix F.1).

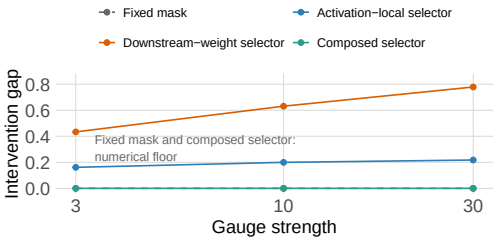
4.3 Pruning decisions and downstream consequences

4.3.1 Procedure dependence

Passing the Qwen gauge to four local pruning procedures gives different outcomes (Table 5, Appendix D.4). Wanda (Sun et al., 2024) is near-invariant because its weight–activation score cancels reciprocal scaling. Magnitude and SparseGPT are more sensitive; balancing reduces SparseGPT's discrepancy. Common-frame mask and weight comparisons, together with FP64 controls, distinguish scalar-damping non-covariance from reconstruction roundoff (Appendix E).



(a) Harmless coordinate relabeling. Same exact dense function.



(b) Genuine representation dependence. Same exact dense function.

Figure 1: The audit separates harmless coordinate disagreement from genuine representation-dependent intervention changes. In (a), inverse transport restores agreement after a hidden-coordinate permutation. In (b), independently recomputed local selectors change the intervention despite preserving the same exact dense function. Panel (b) plots arithmetic means; the text reports pooled medians.

4.3.2 Layer allocation can change while final quality remains close

We extend the procedure audit to deciding where to prune. A common allocator minimizes isolated-layer validation costs under 50% mask sparsity across all MLP down projections, then reconstructs the full sparse model and evaluates test negative log-likelihood (NLL) on held-out WikiText-2 (Merity et al., 2016). SparseGPT* denotes our adaptation that prunes exactly the requested number of weights per block (Appendix H); the allocator is our wrapper. Allocations change in all eight gauge/calibration comparisons per model for native SparseGPT*, while Wanda and balanced SparseGPT* preserve theirs. TinyLlama exceeds the prespecified five-percentage-point mean allocation-change threshold in four of eight comparisons; Qwen exceeds it in none.

The consequence is not determined by the allocation change alone. Let $L(g, a)$ be test NLL after reconstruction in representation g using allocation a. We separately reconstruct the gauge model with the base allocation a_b and with its newly selected allocation a_g. In 15/16 native comparisons, $L(g, a_g) < L(g, a_b)$: the new allocation offsets part or all of the fixed-allocation reconstruction gap. Figure 2 shows the first calibration sample and function-preserving rescaling in the evaluation order for each model. The Qwen example changes NLL from 2.738009 to 2.744571 under fixed allocation, then to 2.737692 with the new allocation.

Thus similar final NLL can coexist with different layer-budget judgments. An importance interpretation of an allocation should be audited jointly with reconstruction and final quality. The ordered diagnostic is reference-dependent, not a unique causal decomposition; the full grid and the small native/balanced method-order reversal are in Appendix H.

4.3.3 Consequence: sparse model to behavior

Let D_{top1} denote the fraction of 848 valid positions at which two models predict different highest-probability next tokens from the same prefix. Superscripts dense/sparse specify the unpruned/pruned base–gauge pair.

The dense pair has $D_{\text{top1}}^{\text{dense}} = 0$. Independent pruning produces $D_{\text{top1}}^{\text{sparse}} = 0.0519/0.0318$ for the two checkpoints, with mean Jensen–Shannon divergence $0.00449/0.00200$. GSM8K (Cobbe et al., 2021) uses eight-shot prompts, greedy 4,096-token decoding, and explicit final answers; strict scoring counts unresolved responses as incorrect. The pooled median strict disagreement is

$$D_{\text{corr}} = 0.0883$$

for the independent sparse pair. Thus, independent pruning maps the same dense solution to different sparse functions and, consequently, to different behavior, as shown in Figure 3. The selector experiment establishes direct selector-to-intervention propagation; the pruning and downstream experiments are separate procedure-level tests. Calibration and implementation checks retain representation dependence, with setting-dependent magnitudes; capped-text and genuinely fixed-token studies are distinguished in Appendix D.6.

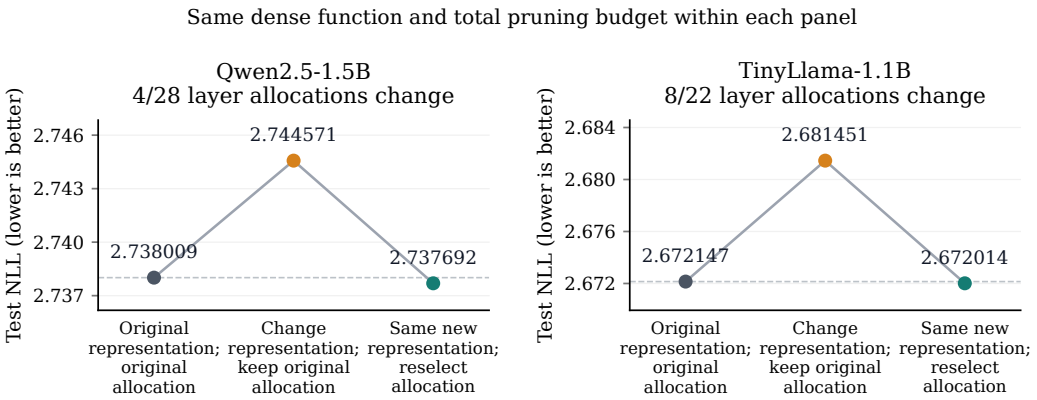
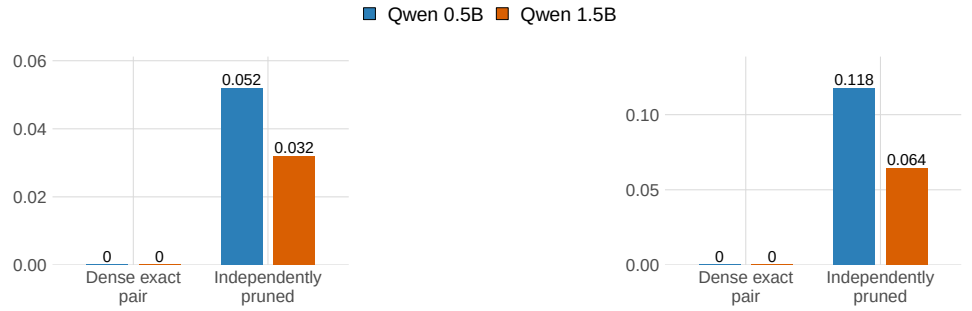


Figure 2: **Close final NLL can hide different layer allocations.** Changing representation while keeping the allocation fixed raises post-pruning NLL (left to middle). Reselecting the allocation on validation data offsets this gap (middle to right), leaving close endpoint NLL despite 4/28 and 8/22 changed layer budgets. Within each panel, the dense function and 50% mask budget in MLP down weights are unchanged. SparseGPT* denotes our adaptation enforcing the exact requested pruned-weight count (Appendix H). Both panels use the first calibration data sample and function-preserving rescaling in the evaluation order. Across both models, reselection lowers NLL relative to fixed allocation in 15 of 16 comparisons.



(a) Token top–1 disagreement.

(b) GSM8K strict correctness disagreement.

Figure 3: Downstream consequence of independently applied pruning. Dense exact-equivalent pairs agree at both token and task interfaces, whereas independently pruned pairs show token-level and paired GSM8K correctness disagreement. Token values are aggregate rates over the evaluation set; task values are strict-score medians over ten nested conditions per checkpoint. Final-answer coverage is 91.1–97.9%; all 20 independent conditions also differ on jointly resolved numeric answers.

4.4 Transformation-aware repair improves consistency

Composing reciprocal quantities, $a_i' b_i' = a_i b_i$, restores selector Jaccard from approximately 0.2995 to 1. Balancing before SparseGPT and transporting its output back raises method-grid mask Jaccard from 0.834/0.830 to 0.988/0.978 and lowers pooled strict GSM8K disagreement from 0.0883 to 0.0299 (Figure 4, Appendix D.7). Strict disagreement decreases in all 20 task conditions. Mean correctness-reduction bounds include zero for 1.5B; numeric-answer reduction bounds are positive for both checkpoints (Appendix D.11). Consistency repair and task utility remain separate outcomes.

4.5 Edit acceptance beyond rewrite success

We adapt constrained fine-tuning (FT) and Rank-One Model Editing (ROME; Meng et al., 2022) to 200 held-out CounterFact requests per model, screened for single-token targets and an initial old-over-new preference. Acceptance requires rewrite success, success on at least 80% of supplied paraphrases, and no reversal of an originally preferred neighborhood answer. These criteria compare

Table 1: Edit-acceptance disagreement (%). Each request averages four gauge/base comparisons; $n = 200$ per model. Brackets are simultaneous 95% intervals for the 62-estimate editing family, including zero-event bounds. The two control methods each have zero observed disagreement.

Checkpoint	Native FT	Balanced FT / ROME
Qwen 1.5B	8.125 [2.337, 13.913]	0 [0, 3.833]
TinyLlama 1.1B	7.875 [1.869, 13.881]	0 [0, 3.833]

old/new target likelihoods. Native FT makes the new rewrite target top-1 for every request and representation, yet acceptance differs by 8.125%/7.875% (Table 1); both differences survive the three-application correction.

Fixed scalar constraints and coordinatewise optimization define different functional edits after rescaling. The diagnosis is to specify the editor's coordinate convention and audit the complete acceptance verdict alongside utility. Balanced FT and ROME have zero observed disagreement, but balanced FT's two-percentage-point utility noninferiority is not established across all conditions. Cohort restrictions, the declared acceptance rule, and transport controls are detailed in Appendix I.

The complementary BLiMP grammatical minimal-pair audit (Warstadt et al., 2020) finds changed activation masks but a fixed calibration-selected top candidate layer (Appendix G.4). Selector instability therefore need not change the reported decision.

4.6 Using EFSS in research

Table 2: Use EFSS when a conclusion should hold across exact-equivalent representations.

Research task	Audit question	What to compare	Use of the result
Analysis design. Attribution or importance methods.	Does the output characterize the function?	Scores, rankings or explanations in a common frame.	Qualify the claim or revise the observable.
Procedure design. Pruning, editing or intervention.	Where does coordinate dependence enter?	Intermediate choices, output functions and utility.	Declare coordinates or repair the affected step.
Benchmarking. Comparing methods.	Does the comparison depend on representation?	Paired utility differences, method order and verdicts.	Qualify the ranking or acceptance claim.

Apply the audit: (1) Specify the claim, domain and output comparison. (2) Choose analytically certified transformations and check numerics and transport. (3) Fix data and budgets, declare constraints, and rerun independently. (4) Compare aligned outputs, downstream decisions and utility. Qualify representation-dependent claims or repair the procedure when appropriate; coordinate dependence can be intentional. A pass establishes consistency only on the tested family. Worked examples: Appendices H.2, I.4, and G.4.

5 Conclusion

EFSS identifies neural solutions by their realized functions and tests scientific claims under their declared Output Semantics. Controls, interventions, and pruning studies establish that representation dependence can reach downstream behavior. The application audits make its uses concrete: the same dense function can lead to different pruning allocations and edit-acceptance verdicts, while unstable unit selections can preserve a coarser layer choice.

The practical requirement is to declare the claim, transformation family, transport, and procedure constraints, then assess that claim alongside intermediate outputs and task utility. Representative alignment can improve consistency without guaranteeing better task performance. These are certified-pair audits of the tested families, not guarantees over complete functional fibers.

AI USE STATEMENT

Generative AI tools were used for literature discovery, discussion and refinement of the conceptual framing and experimental design, assistance with checking mathematical derivations and proofs, analysis and interpretation of experimental results, drafting and editing manuscript text, and visualization of experimental data. AI-generated numerical assertions were not treated as experimental evidence. The authors take full responsibility for the manuscript and its contents.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2312.11983.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2401.15024.

Amir Asiaee. Causal mechanism reduction: Mechanism replacement for neural network pruning and abstraction. *arXiv preprint arXiv:2602.24266*, 2026.

Joachim Bona-Pellissier, François Bachoc, and François Malgouyres. Parameter identifiability of a deep feedforward relu neural network. *Machine Learning*, 112(11):4431–4493, 2023. doi: 10.1007/s10994-023-06355-4. URL https://arxiv.org/abs/2112.12982.

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

Hang-Cheng Dong and Pengcheng Cheng. Quotient geometry, effective curvature, and implicit bias in simple shallow neural networks. *arXiv preprint arXiv:2603.21502*, 2026.

Matthew Farrugia-Roberts. Functional equivalence and path connectivity of reducible hyperbolic tangent networks. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, 2023.

Andrea Gesmundo and Kaitlin Maile. Composable function-preserving expansions for transformer architectures. *arXiv preprint arXiv:2308.06103*, 2023.

J. Elisenda Grigsby, Kathryn Lindsey, and David Rolnick. Hidden symmetries of relu networks. In *International Conference on Machine Learning*, pp. 11734–11760, 2023.

J. Elisenda Grigsby, Kathryn Lindsey, Robert Meyerhoff, and Chenxi Wu. Functional dimension of feedforward relu neural networks. *Advances in Mathematics*, 482:110636, 2025. doi: 10.1016/j.aim.2025.110636.

Moritz Grillo and Guido Montúfar. Most relu networks admit identifiable parameters. *arXiv preprint arXiv:2605.03601*, 2026.

Tianteng Gu, Bei Liu, Bo Xiao, Ke Zeng, Jiacheng Liu, and Yanmin Qian. DenoiseRotator: Enhance pruning robustness for LLMs via importance concentration. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.23049.

Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 5, 1993. URL https://proceedings.neurips.cc/paper/1992/hash/303ed4c69846ab36c2904d3ba8573050-Abstract.html.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529, 2019.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, 1990. URL https://proceedings.neurips.cc/paper/1989/hash/6c9882bbac1c7093bd25041881277658-Abstract.html.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. URL https://arxiv.org/abs/1609.07843.

Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant architectures for learning in deep weight spaces. In *International Conference on Machine Learning*, 2023.

Mary Phuong and Christoph H. Lampert. Functional vs. parametric equivalence of relu networks. In *International Conference on Learning Representations*, 2020.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI technical report*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Pranavkrishnan Ramakrishnan. A complete symmetry classification of shallow relu networks. *arXiv preprint arXiv:2604.14037*, 2026.

Pierre Stock and Rémi Gribonval. An embedding of relu networks and an analysis of their identifiability. *Constructive Approximation*, 57(2):853–899, 2023. doi: 10.1007/s00365-022-09578-1. URL https://arxiv.org/abs/2107.09370.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.11695.

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

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

Viet-Hoang Tran, Thieu N. Vo, An Nguyen The, Tho Tran Huu, Minh-Khoi Nguyen-Nhat, Thanh Tran, Duy-Tung Pham, and Tan Minh Nguyen. Equivariant neural functional networks for transformers. In *International Conference on Learning Representations*, 2025.

Verner Vlačić and Helmut Bölcskei. Affine symmetries and neural network identifiability. *Advances in Mathematics*, 376:107485, 2021. doi: 10.1016/j.aim.2020.107485.

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *International Conference on Machine Learning*, 2019.

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.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. BLiMP: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392, 2020. doi: 10.1162/tacl_a_00321. URL https://aclanthology.org/2020.tacl-1.25/.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099, 2023.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

Bo Zhao, Robin Walters, and Rose Yu. Symmetry in neural network parameter spaces. *Transactions on Machine Learning Research*, 2025.

A Extended Related Work and Positioning

This appendix places the paper question in relation to prior work on functional equivalence, representation comparison, reparameterization-aware analysis, and model compression. The relevant distinction is not whether these literatures have studied symmetries or invariance; they have. The distinction is the level at which a scientific claim is defined and tested.

A.1 Functional identity, realization fibers, symmetries, and identifiability

Neural-network identifiability relates parameters to the functions they realize, accounting for affine symmetries of the nonlinearity (Vlačić & Bölcskei, 2021). For ReLU networks, Phuong and Lampert analyzed functional versus parametric equivalence and showed how permutation and positive-rescaling operations account for equivalence in specified architectures (Phuong & Lampert, 2020). Stock and Gribonval developed a scaling-invariant embedding and local identifiability conditions, while Bona-Pellissier et al. gave conditions for recovering deep ReLU parameters modulo permutation and positive rescaling (Stock & Gribonval, 2023; Bona-Pellissier et al., 2023).

More recent work makes the geometry of the equivalence class itself explicit. Hidden symmetries and functional dimension show that the realization map can have nontrivial, parameter-dependent redundancy (Grigsby et al., 2023; 2025). Further studies characterize shallow-network symmetries and reducible or identifiable parameter sets (Ramakrishnan, 2026; Farrugia-Roberts, 2023; Grillo & Montúfar, 2026). Complementary work surveys parameter-space symmetries (Zhao et al., 2025) and studies quotient geometry on regular parameter sets (Dong & Cheng, 2026). These results establish the structural foundation used here: distinct parameterizations may represent one function, and the complete set of such parameterizations is a fiber of the realization map.

Our use of this foundation is deliberately narrower than a new identifiability theorem. For a known function-preserving group G, prior structural results motivate the inclusion $G \cdot \theta \subseteq M_{f_\theta}$; equality requires an additional completeness result and is not assumed in this paper. EFSS uses the ordinary fiber and quotient construction to name the exact functional solution represented by that fiber. The added question is scientific rather than set-theoretic: if an analysis is computed at one representative, when is its reported conclusion well defined on the corresponding solution? Thus the fiber/quotient language organizes established structural facts into the criterion audited in Sections 2 and 3.

A.2 Implementation invariance, representation comparison, and equivariance

Implementation Invariance was introduced as an axiom for input-attribution methods: functionally equivalent implementations should receive the same attribution under the method's declared conditions (Sundararajan et al., 2017). This is closely related to the present concern, but its object is an attribution of an input feature for a fixed prediction. Representation-comparison methods address a different but complementary problem. CKA and related measures compare activation spaces

across layers or models and make explicit that useful similarity notions have particular invariance properties (Kornblith et al., 2019). They do not, by themselves, decide how a labeled selector, an intervention, or a newly produced model should be compared.

Equivariant neural functionals make the transformation law explicit when a model is treated as an input to another neural network. Permutation-equivariant weight-space architectures encode the fact that reordering hidden units leaves the represented function unchanged (Navon et al., 2023). Transformer-NFN work characterizes symmetry actions for attention modules, gives same-function conditions for those modules, and constructs functionals equivariant to the resulting action (Tran et al., 2025). These works provide important principles for building transformation-compatible maps. Our audit asks a complementary question about analyses of a fixed trained model: after exact functional identity is certified, should the output be equal as a scalar, transported as a labeled or structured object, or compared through the function of a model produced by the procedure? We call that choice the Output Semantics. The contribution is therefore not a new invariance axiom or equivariant architecture, but a common output-aware audit across these output types.

A.3 Reparameterization-aware analyses and interventions

Parameter alignment and re-basing methods demonstrate that coordinate correspondence can matter even when the represented computation is unchanged. Git Re-Basin, for example, aligns permutation-related neurons before merging models, treating permutation symmetry as an operational issue for parameter-space comparison (Ainsworth et al., 2023). Such methods provide useful transformation-specific correspondences and positive controls for the labeled-output case.

Intervention and causal-abstraction work studies what happens when internal variables or mechanisms are changed. Causal Mechanism Reduction (CMR) is especially close to the present setting: it treats a network as a deterministic structural causal model, studies mechanism replacement, and explicitly tests a positive-scaling invariance requirement for reduction criteria (Asiaee, 2026). This is important prior art for reparameterization-aware intervention design. The present paper does not claim that such transformation-aware scores or interventions are new. It uses exact-equivalent pairs to organize several questions under one audit: whether a local selector changes, whether that change propagates to an intervention, and whether the prescribed output correspondence removes the difference. The Qwen experiment supplies one concrete architecture-level instance of this broader audit.

A.4 Pruning and model-producing procedures

Pruning is a model-producing analysis: it maps a dense parameterization to a new, sparse model. Classical work uses saliency or curvature to decide which parameters to remove, while Wanda and SparseGPT provide widely used one-shot procedures for large language models (LeCun et al., 1990; Hassibi & Stork, 1993; Sun et al., 2024; Frantar & Alistarh, 2023). The question here is not which method has the best compression–accuracy trade-off. It is whether applying the same procedure independently to two exact-equivalent representatives yields corresponding sparse structures and, ultimately, the same resulting function.

Several neighboring methods show why this question deserves an explicit comparison. EigenDamage changes the basis in which a Hessian-based pruning decision is made, making the coordinate system of the pruning rule an explicit design choice (Wang et al., 2019). SmoothQuant uses a compensated activation–weight rescaling to move quantization difficulty while preserving the dense computation (Xiao et al., 2023). Function-preserving expansion methods for Transformers likewise construct exact transformations across parameterizations with different architecture sizes (Gesmundo & Maile, 2023). These works provide reparameterization tools or procedure-specific objectives, but they do not replace the need to specify the output relation when the procedure itself returns a new model.

CMR provides the closest recent overlap on pruning under exact positive scaling: its reduction framework reports representation sensitivity for variance-based pruning and invariance for a logit-distortion score (Asiaee, 2026). Our scope is different. We compare several existing pruning procedures on paired exact-equivalent dense models, transport masks and surviving weights under the declared correspondence, and then trace procedure-level differences to token- and task-level out-

puts. The resulting claim is bounded to the tested transformation family and procedures; it is not a universal ranking of pruning methods or a new pruning objective.

SliceGPT exploits computational invariance to choose compression coordinates, and DenoiseRotator concentrates importance through learned rotations before pruning (Ashkboos et al., 2024; Gu et al., 2025). These establish closely related uses of function-preserving coordinates. Here the common audit links the meaning of the reported output to a concrete diagnosis: selector-induced interpretation changes (Appendix F.1), a coordinate-dependent reconstruction penalty, or numerical residuals after exact orbit alignment (Appendix E). This supplies evidence for interpreting and repairing the tested analyses, alongside the existing positive controls.

A.5 Model editing and acceptance decisions

Model editing also returns a new model. ROME studies factual associations through a rank-one editing construction and CounterFact evaluations of rewriting, generalization, and neighborhood specificity (Meng et al., 2022). We use these evaluation dimensions in a declared acceptance rule. The additional question is whether the acceptance judgment depends on the original dense function alone or on the editor's coordinate convention. Constrained FT and a covariant ROME adaptation provide contrasting outcomes (Appendix I). This adds a use of the common audit, not a new editing algorithm.

A.6 Positioning summary

Prior work establishes the ingredients: functional equivalence and identifiability characterize the many-to-one realization map; symmetry and equivariance work supplies transformation laws; representation and intervention methods provide local comparison principles; and pruning/quantization work supplies procedures that can act on reparameterized models. This paper connects those ingredients at the level of solution-level validity.

Specifically, we define exact sameness by equality of the realized function, distinguish a known symmetry orbit from the complete fiber, declare an output-specific comparison rule, and test existing analyses on exact-equivalent pairs. The empirical contribution traces representation dependence across intermediate outputs, reported decisions, and utility: changed layer allocations can compensate reconstruction differences, edit acceptance can change despite stable rewrite success, and unstable selectors can preserve the selected top candidate layer. These distinctions make the common audit useful for interpreting a reported result and identifying the stage that needs qualification or repair.

Table 3 separates the shared ingredients from each work's primary purpose.

B Exact Functional Neural Solutions

This appendix follows Section 2 subsection by subsection. It provides the proofs, worked examples, and scope clarifications supporting the definitions of exact functional solutions.

B.1 From Parameterizations to Exact Functional Identity

This subsection expands the realization map and exact-equivalence relation used in Section 2.1. All equalities below are understood on the declared input domain and at the declared output interface.

Linear realization. For a linear architecture with input dimension d and output dimension k,

$$\Theta_A = \mathbb{R}^{k\times d} \times \mathbb{R}^k, \qquad R_A(W,b)(x) = Wx + b.$$

The parameterization is the pair (W, b), whereas the realized object is the function $x \mapsto Wx + b$.

Two-layer ReLU realization. For a two-layer network,

$$R_A(W_1, W_2)(x) = W_2\sigma(W_1 x),$$

where σ is coordinatewise ReLU. The realization map therefore hides the intermediate coordinates when it returns the input–output function.

Table 3: Positioning EFSS relative to representative prior work. Symbols compare the stated scope, rather than performance or guarantees over the complete functional fiber.

Representative work	Exact identity	Invariant output	Output transport	Produced model	Primary purpose
Functional-equivalence theory (Phuong & Lampert, 2020; Grigsby et al., 2023)	✓	n/a	n/a	n/a	Characterize equivalent parameterizations and hidden symmetries.
Implementation Invariance (Sundararajan et al., 2017)	✓	✓	n/a	n/a	Attribute input features consistently across equivalent implementations.
CKA (Kornblith et al., 2019)	–	✓	n/a	n/a	Compare representations, including those of different learned functions.
DWSNets (Navon et al., 2023)	✓	✓	✓	✓	Learn invariant predictions and equivariant weight updates.
Transformer-NFN (Tran et al., 2025)	✓	✓	✓	n/a	Build symmetry-compatible weight-space networks for performance prediction.
CMR (Asiaee, 2026)	✓	✓	–	✓	Unify mechanism replacement, pruning, and causal-abstraction verification.
SmoothQuant / SliceGPT (Xiao et al., 2023; Ashkboos et al., 2024)	✓	–	–	✓	Choose function-preserving coordinates to improve compression.
ROME (Meng et al., 2022)	–	–	–	✓	Rewrite factual associations and evaluate edit quality.
EFSS (this work)	✓	✓	✓	✓	**Audit existing analyses across output types; trace interpretation, allocation, and edit-acceptance decisions.**

Symbols. ✓: explicitly addressed; –: not an explicit focus; n/a: no output of this type is studied. A dash does not mean failure.

Columns. *Exact identity*: functional equivalence is a stated object, axiom, symmetry, or tool. *Invariant output / output transport*: explicit rules for analysis outputs under declared transformations, with nontrivial correspondence in the latter case. *Produced model*: a returned model is evaluated; pairwise consistency is a separate question. CMR's abstraction maps are not the reparameterization-output maps counted here.

Transformer realization. For a Transformer parameterization θ and a declared sequence domain X, $R_A(\theta)$ is the corresponding input-to-logit function. If the output interface is instead defined as probabilities or decoded tokens, that choice changes the function being compared and must be fixed before asserting exactness.

Scalar rescaling. The simplest exact pair is

$$h = 2x, \quad y = 3h = 6x \qquad \text{and} \qquad h' = 4x, \quad y' = 1.5h' = 6x.$$

The hidden value and downstream coefficient differ, but the two parameterizations agree for every x.

Positive-homogeneous rescaling. For ReLU and any $c > 0$,

$$v\,\sigma(w^\top x) = \frac{v}{c}\,\sigma\big((cw)^\top x\big)\,.$$

Thus (w, v) and $(cw, v/c)$ are distinct parameterizations with the same realized function on all of X.

Hidden-unit permutation. Consider

$$f_\theta(x) = \sum_{j=1}^{m} v_j \, \sigma(w_j^\mathsf{T} x).$$

For any permutation π of $\{1, \ldots, m\}$, define θ' by reordering the hidden units. Reindexing the sum gives $f_{\theta'}(x) = f_\theta(x)$ for every x, even though the parameter coordinates and neuron labels have changed.

Linear factorization. For a two-layer linear network with $f_\theta(x) = W_2 W_1 x$ and any invertible hidden-coordinate matrix S,

$$f_{\theta'}(x) = (W_2 S^{-1})(S W_1)x = W_2 W_1 x = f_\theta(x).$$

This is an exact example of factorization redundancy.

Step-by-step verification of Eq. (2). The relation $\equiv$ is an equivalence relation:

1. *Reflexivity:* $f_\theta(x) = f_\theta(x)$ for every $x \in X$, so $\theta \equiv \theta$.
2. *Symmetry:* if $f_\theta(x) = f_\eta(x)$ for every x, equality can be reversed, so $\eta \equiv \theta$.
3. *Transitivity:* if $f_\theta(x) = f_\eta(x)$ and $f_\eta(x) = f_\zeta(x)$ for every x, then $f_\theta(x) = f_\zeta(x)$, so $\theta \equiv \zeta$.

The requirement is equality for every input in X. A finite test set or a small numerical residual can support an implementation check, but it does not replace this all-domain condition.

B.2 Functional Fibers and the Exact Solution Space

This subsection expands the fiber, quotient, and quotient–realization construction with concrete examples. Unless stated otherwise, all examples use one fixed architecture and one fixed output interface.

Example 1: a scalar linear fiber. Consider the two-layer scalar architecture

$$f_{a,b}(x) = ba\,x, \qquad \Theta_A = \mathbb{R}^2.$$

The function $f(x) = 6x$ has the fiber

$$M_{6x} = \{(a, b) \in \mathbb{R}^2 : ba = 6\}.$$

Thus $(2, 3)$, $(6, 1)$, and $(-2, -3)$ are different parameter points but belong to the same exact functional solution. The fiber is an entire curve in parameter space, not a single preferred point. In particular,

$$q_A(2, 3) = q_A(6, 1) = q_A(-2, -3).$$

Example 2: positive-homogeneous ReLU scaling. For a scalar ReLU unit with output coefficient v,

$$f_{w,v}(x) = v\,\sigma(wx),$$

where σ is ReLU. For any $c > 0$, positive homogeneity gives

$$v\,\sigma(wx) = \frac{v}{c}\,\sigma(cwx).$$

For example, $(w, v) = (1, 2)$ and $(w', v') = (4, 0.5)$ both realize $2\sigma(x)$. They are different points in Θ_A but lie in the same fiber. The quotient identifies them without requiring a particular scaling representative.

Example 3: hidden-unit permutation. Consider the two-unit network

$$f_\theta(x) = 2\sigma(x) + 3\sigma(-x).$$

A parameterization that stores the first unit before the second unit and a parameterization that stores them in the reverse order have different coordinate arrays and different neuron labels. Nevertheless, addition is commutative, so both realize the same function for every x. The two arrays therefore represent one element of $\Theta_A/\equiv$. This is why a raw coordinate comparison can disagree even when the quotient element is unchanged.

Example 4: matrix-factorization fiber. For a two-layer linear network with $x \in \mathbb{R}^2$, let

$$f_\theta(x) = W_2 W_1 x, \qquad W_1 = I_2, \qquad W_2 = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}.$$

Choose

$$S = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad S^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}.$$

The transformed factors are

$$W_1' = SW_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad W_2' = W_2 S^{-1} = \begin{bmatrix} 2 & -2 \\ 0 & 3 \end{bmatrix}.$$

Although both matrices change,

$$W_2' W_1' = W_2 S^{-1} S W_1 = W_2 W_1,$$

so the two parameterizations lie in the same fiber. The quotient keeps the realized linear map and forgets which hidden-coordinate factorization was used.

Example 5: duplicated and inactive hidden channels. Use the same two-unit architecture

$$f_\theta(x) = v_1 \sigma(w_1 x) + v_2 \sigma(w_2 x).$$

The parameterizations

$$\theta: \quad (w_1, v_1) = (1, 1), \quad (w_2, v_2) = (1, 1)$$

and

$$\theta': \quad (w_1', v_1') = (1, 2), \quad (w_2', v_2') = (7, 0)$$

both realize

$$f(x) = 2\sigma(x).$$

The second representation contains an inactive channel whose incoming weight can be changed without changing the function. This example is not merely a relabeling of the first parameterization; it shows why a complete fiber can be larger than one familiar symmetry orbit.

Example 6: distinct functions have disjoint fibers. The functions $f(x) = 6x$ and $g(x) = 2x$ have different fibers:

$$M_f = \{(a, b) : ba = 6\}, \qquad M_g = \{(a, b) : ba = 2\}.$$

They cannot share a parameter point. If (a, b) belonged to both sets, then the same parameterization would have to realize both $6x$ and $2x$ on every input, which is impossible. Hence

$$f \neq g \quad \Longrightarrow \quad M_f \cap M_g = \varnothing.$$

This gives the partition property in a concrete form.

Representative table. The preceding examples can be summarized as follows:

Parameter representative	Realized function	Quotient element
$(2, 3)$	$6x$	$[\theta]_\equiv$
$(6, 1)$	$6x$	$[\theta]_\equiv$
$(-2, -3)$	$6x$	$[\theta]_\equiv$
$(1, 1)$ and $(1, 1)$	$2\sigma(x)$	$[\phi]_\equiv$
$(1, 2)$ and $(7, 0)$	$2\sigma(x)$	$[\phi]_\equiv$

The table emphasizes that the quotient column records exact functional identity, not the number of parameter arrays or the internal decomposition used to produce the function.

Fiber and equivalence class. For completeness, let $f \in \mathcal{F}_A$ and choose θ with $f_\theta = f$. If $\eta \in [\theta]_\equiv$, then $\theta \equiv \eta$, so $f_\eta = f_\theta = f$ and $\eta \in M_f$. Conversely, if $\eta \in M_f$, then $f_\eta = f = f_\theta$, so $\theta \equiv \eta$ and $\eta \in [\theta]_\equiv$. Therefore

$$[\theta]_\equiv = M_{f_\theta}.$$

This equality is what turns the informal phrase "all representations of the same solution" into a precise set.

Quotient–realization bijection. The induced map is

$$\widetilde{R}_A([\theta]_{\equiv}) = f_\theta.$$

It is well-defined: if $[\theta]_{\equiv} = [\eta]_{\equiv}$, then $\theta \equiv \eta$ and $f_\theta = f_\eta$. It is injective: if $\widetilde{R}_A([\theta]_{\equiv}) = \widetilde{R}_A([\eta]_{\equiv})$, then $f_\theta = f_\eta$, hence $\theta \equiv \eta$ and the classes are equal. It is surjective because every $f \in \mathcal{F}_A$ is, by definition, $R_A(\theta)$ for some $\theta \in \Theta_A$.

An explicit inverse is

$$\Psi : \mathcal{F}_A \longrightarrow \mathcal{S}_A^{\mathrm{ex}}, \qquad \Psi(f) = M_f.$$

This is well-defined because M_f is exactly one equivalence class whenever $f \in \mathcal{F}_A$. Thus $\Psi = \widetilde{R}_A^{-1}$ as a set-level map. No arbitrary representative must be selected to define the inverse.

What the quotient does and does not provide. The quotient does not choose $(2, 3)$ instead of $(6, 1)$, does not produce a canonical parameter vector, and does not automatically provide a computational procedure for recovering a representative. It performs a set-level identification: all parameterizations in one fiber become one exact functional solution. Canonicalization, optimization, or a code-level representative is an additional construction. Likewise, the quotient is defined only after the architecture, full input domain, and output interface have been fixed. It does not identify parameterizations from different system specifications merely because they agree on a finite collection of inputs.

B.3 A Unified Language for Neural Solution Identity

This subsection gives concrete certificates for several familiar transformations. The purpose is not to assume that every exact equivalence is generated by one of these transformations. Rather, each calculation provides a verifiable family of points inside one functional fiber and shows how known symmetry questions fit the EFSS language.

Permutation certificates. Consider a two-layer network with hidden width m,

$$f_{W_1,W_2}(x) = W_2\,\sigma(W_1 x),$$

where σ acts coordinatewise. Let P be an $m \times m$ permutation matrix and define

$$W_1' = PW_1, \qquad W_2' = W_2P^{-1}.$$

Because coordinatewise nonlinearities commute with relabeling, $\sigma(Pz) = P\sigma(z)$. Therefore, for every input x,

$$\begin{aligned} f_{W_1',W_2'}(x) &= W_2P^{-1}\sigma(PW_1x) \\ &= W_2P^{-1}P\sigma(W_1x) \\ &= f_{W_1,W_2}(x). \end{aligned}$$

The hidden units have been renamed, but the realized function has not changed. A raw list of channel indices can consequently change under P, while a list transported by P^{-1} can agree exactly. This is why a labeled output is not compared by raw array equality alone.

Positive-scaling certificates. For a ReLU hidden layer, let $D = \mathrm{diag}(c_1, \ldots, c_m)$ with $c_j > 0$. The transformation

$$W_1' = DW_1, \qquad W_2' = W_2D^{-1}$$

preserves the function. Indeed, positive homogeneity gives $\sigma(Dz) = D\sigma(z)$, and hence

$$W_2'\sigma(W_1'x) = W_2D^{-1}\sigma(DW_1x) = W_2D^{-1}D\sigma(W_1x) = W_2\sigma(W_1x).$$

The scalar instance makes the mechanism explicit: $(w, v) = (1, 2)$ and $(w', v') = (4, 1/2)$ both compute $2\sigma(x)$. The hidden activation is four times larger, while its downstream coefficient is four times smaller.

Reciprocal channel gauges. The same cancellation can be written at the level of one channel. If a channel contributes $v_j h_j(x)$ and $c_j \neq 0$, then

$$h'_j(x) = c_j h_j(x), \qquad v'_j = v_j / c_j$$

gives $v'_j h'_j(x) = v_j h_j(x)$ for every x. For a collection of channels, the scalar c_j becomes a nonsingular diagonal map. In an actual architecture, this certificate is valid only when the upstream and downstream transformations are lifted consistently through the declared module and output interface. The calculation identifies the exact cancellation; the model-specific implementation must still verify that no intervening operation breaks it.

Factorization certificates. For a linear map written as $W = W_2 W_1$ with an intermediate width m, every $S \in \mathrm{GL}(m)$ gives

$$W'_1 = S W_1, \qquad W'_2 = W_2 S^{-1},$$

and therefore $W'_2 W'_1 = W_2 W_1$. For example, with

$$W_1 = I_2, \quad W_2 = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},$$

we obtain

$$W'_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad W'_2 = \begin{bmatrix} 2 & -2 \\ 0 & 3 \end{bmatrix}, \qquad W'_2 W'_1 = W_2 W_1.$$

The internal factorization changes, but the end-to-end linear function is identical. This example also shows why a weight-space property need not be a property of the composed map.

Orbit inclusion. Let a group G act on Θ_A and suppose that each action is function-preserving:

$$R_A(g \cdot \theta) = R_A(\theta) \qquad \forall g \in G.$$

Then every point in the orbit $G \cdot \theta$ belongs to the fiber M_{f_θ}. In symbols,

$$G \cdot \theta \subseteq M_{f_\theta}.$$

The proof is immediate but important: an orbit point has the form $g \cdot \theta$ for some g, and the assumed certificate gives $f_{g\cdot\theta} = f_\theta$. Thus a known symmetry supplies a certified exact-equivalent family. It equals the full fiber only if every exact-equivalent parameterization can be reached by the group action; the residual cases are discussed in Appendix B.4.

Identifiability and canonicalization. In the scalar model $f_{a,b}(x) = ba\,x$, restrict first to $a, b > 0$ and fix the function $6x$. The positive-scaling orbit of $(2, 3)$ is

$$\{(2c, 3/c) : c > 0\} = \{(a, b) : a > 0,\ b > 0,\ ab = 6\} = M_{6x} \cap \mathbb{R}^2_{>0}.$$

Within this restricted domain, the known scaling family describes the entire fiber. A consistent representative can also be selected by

$$C(a, b) = (\sqrt{ab}, \sqrt{ab}),$$

which maps every positive point in the fiber of $6x$ to $(\sqrt{6}, \sqrt{6})$. This is a canonicalization on the positive branch, not a claim that one universal canonicalizer exists for every architecture or every fiber. If the domain is enlarged to all real (a, b), the negative branch $(-2, -3)$ is also in M_{6x} and is not reached by positive scaling. The example separates three questions: whether a transformation is exact, whether it spans the full fiber, and whether a representative can be selected consistently.

Summary of the certificates. Permutations, positive scalings, reciprocal gauges, and factorization changes are different mechanisms, but each is checked by the same EFSS test: evaluate the transformed parameterization on the declared input domain and verify equality of the realized function. Once this equality is established, the transformation belongs to a certified representation family within one solution. Appendix B.4 gives examples where the certified family is strictly smaller than the full fiber.

B.4 Beyond Known Symmetries: Residual Exact Structure

This subsection gives concrete examples for the distinction between a known symmetry orbit and the complete exact fiber.

The general residual-quotient test. Suppose a group G acts on Θ_A and preserves the realization map. For any $\theta \in \Theta_A$, the orbit is contained in the exact fiber:

$$G \cdot \theta \subseteq M_{f_\theta}.$$

If the inclusion is strict, then there exists an exact-equivalent parameterization $\eta \in M_{f_\theta}$ that is not reachable from θ by the declared symmetry. Consequently, quotienting by G does not collapse the whole fiber to one point; it leaves at least one additional orbit class. This is the precise sense in which a known symmetry family can remove some representation redundancy without exhausting all exact structure.

The same observation gives a practical audit rule. A pass over a tested G-orbit establishes consistency on that orbit, not automatically on M_{f_θ}. To claim full-fiber invariance, one needs either a theorem that $G \cdot \theta = M_{f_\theta}$ under the declared specification or an additional argument covering the residual classes. The following examples show both discrete and continuous residual structure.

A strict orbit–fiber inclusion. Consider a two-layer scalar linear network

$$f_{a,b}(x) = ba\,x.$$

At $\theta = (2, 3)$, the realized function is $f_\theta(x) = 6x$. Let the known positive-scaling group $G = \mathbb{R}_{>0}$ act by

$$g_c \cdot (a, b) = (ca, b/c), \qquad c > 0.$$

Every point on the orbit of θ realizes the same function:

$$G \cdot \theta = \{(2c, 3/c) : c > 0\} \subseteq M_{6x} = \{(a, b) : ba = 6\}.$$

The inclusion is strict because $(-2, -3) \in M_{6x}$ but $(-2, -3) \notin G \cdot \theta$. Thus the known positive-scaling symmetry explains the positive branch of the fiber, but not the negative branch. This is a concrete instance of

$$G \cdot \theta \subsetneq M_{f_\theta}.$$

Residual quotient after removing a known symmetry. For the same example, the positive branch and the negative branch are each one G-orbit. Therefore the quotient M_{6x}/G contains two elements, represented by $(2, 3)$ and $(-2, -3)$. The quotient by the known scaling removes within-branch redundancy but leaves a discrete residual choice. A pass over the positive-scaling orbit cannot determine how an analysis behaves on the other branch.

A residual family from a null output channel. Consider a two-unit ReLU network

$$f_\theta(x) = v_1\sigma(w_1x) + v_2\sigma(w_2x).$$

If $v_2 = 0$, then changing w_2 to any value w_2' gives

$$v_1\sigma(w_1x) + 0 \cdot \sigma(w_2x) = v_1\sigma(w_1x) + 0 \cdot \sigma(w_2'x) \qquad \forall x.$$

The parameterizations are therefore exactly equivalent even though the incoming weights of the null output channel differ. A standard scaling or permutation family need not connect all such choices, so inactive channels provide a simple source of residual exact structure.

A singular fiber and a differential null direction. Return to $f_{a,b}(x) = ba\,x$ and consider the zero parameterization $(0, 0)$. Its zero-function fiber is

$$M_0 = \{(a, b) : ab = 0\},$$

the union of the two coordinate axes. The differential at $(0, 0)$ is

$$DR_{A,(0,0)}[\delta a, \delta b] = (b\,\delta a + a\,\delta b)x\big|_{(a,b)=(0,0)} = 0$$

for every direction $(\delta a, \delta b)$. Hence the differential nullspace is all of $\mathbb{R}^2$. However, the direction $(1, 1)$ cannot be the velocity of an exact fiber curve through $(0, 0)$. If a differentiable curve had $\gamma'(0) = (1, 1)$, then $a(t) = t + o(t)$ and $b(t) = t + o(t)$, so $a(t)b(t) = t^2 + o(t^2)$, which cannot remain identically zero. This shows concretely why a differential null direction need not integrate into a smooth exact-fiber branch at a singular point.

C Solution-Level Claims and Exact-Functional Audit

This appendix expands Section 3 in the same order as the main text. It gives the factorization proof, concrete output-comparison examples, the audit logic, and the derivations used for consistency repair. Throughout, the architecture, input domain, output interface, and exact transformation family are treated as declared parts of the experiment.

C.1 When Is a Quantity a Property of the Solution?

Section 3 distinguishes a quantity computed from one parameterization from a quantity that is well defined on the exact solution. The distinction is made precise by the quotient map

$$q_A : \Theta_A \longrightarrow \mathcal{S}_A^{\mathrm{ex}}, \qquad q_A(\theta) = [\theta]_{\equiv}.$$

Proof of Proposition 2. Suppose first that H is constant on every exact functional fiber. Define

$$\overline{H}([\theta]_{\equiv}) = H(\theta).$$

This definition is well defined. Indeed, if $[\theta]_{\equiv} = [\eta]_{\equiv}$, then $\theta \equiv \eta$, so the assumed fiber constancy gives $H(\theta) = H(\eta)$. By construction,

$$(\overline{H} \circ q_A)(\theta) = \overline{H}([\theta]_{\equiv}) = H(\theta),$$

and therefore $H = \overline{H} \circ q_A$.

Conversely, suppose that a map $\overline{H}$ satisfies $H = \overline{H} \circ q_A$. If $\theta \equiv \eta$, then $q_A(\theta) = q_A(\eta)$, and hence

$$H(\theta) = \overline{H}(q_A(\theta)) = \overline{H}(q_A(\eta)) = H(\eta).$$

Thus H is constant on every fiber.

Finally, the factor map is unique because every element of $\mathcal{S}_A^{\mathrm{ex}}$ has the form $q_A(\theta)$ for some $\theta \in \Theta_A$. Its value is forced to be $\overline{H}(q_A(\theta)) = H(\theta)$. This proves both directions and uniqueness.

Example: a local quantity versus a solution quantity. For the scalar two-layer model $f_{a,b}(x) = ba\,x$, the parameter a is representation-local. For example, $(a, b) = (2, 3)$ and $(a', b') = (4, 1.5)$ realize the same function, but $a \neq a'$. Therefore $H_{\mathrm{local}}(a, b) = a$ cannot factor through q_A. In contrast,

$$H_{\mathrm{prod}}(a, b) = ab$$

has the same value on the entire fiber of a linear function $f(x) = cx$: $H_{\mathrm{prod}} = c$. It therefore defines a function on the quotient, even though the individual factors do not.

Example: a useful local diagnostic. Failure of factorization does not make H_{local} meaningless. The value of a can still describe the chosen implementation, support a numerical procedure, or help diagnose a particular representation. The mathematical failure is narrower: it cannot be reported without qualification as an intrinsic property of the exact solution.

C.2 Invariance, Equivariance, and Output Semantics

The appropriate comparison depends on what the analysis returns. The following examples make the distinction explicit.

Invariant scalar. Let $s(\theta)$ be a scalar summary, such as a loss evaluated under a fixed declared interface. If $g \cdot \theta \equiv \theta$, a solution-level claim requires

$$s(g \cdot \theta) = s(\theta).$$

For instance, if two parameterizations realize the same logit function on the full declared domain, a scalar defined directly from that function has one value for both representatives. A difference in the scalar is evidence of either a changed interface, a numerical implementation effect, or representation dependence in the reported quantity.

Labeled vector. Suppose an analysis returns one score per hidden channel. Let P permute the channels and let $z(\theta) \in \mathbb{R}^m$ be the score vector. The correct transformation law is

$$z(P \cdot \theta) = Pz(\theta), \qquad \tau_P(z) = Pz.$$

For a concrete two-channel example,

$$z(\theta) = \begin{bmatrix} 0.9 \\ 0.2 \end{bmatrix}, \qquad z(P \cdot \theta) = \begin{bmatrix} 0.2 \\ 0.9 \end{bmatrix}.$$

The raw vectors differ, but inverse transport gives

$$P^{-1} z(P \cdot \theta) = z(\theta).$$

A raw comparison would call this disagreement; a transported comparison recognizes the same scores attached to the same underlying channels.

Transported mask or intervention. Let $m(\theta) \in \{0,1\}^m$ be a selected channel mask. Under the same permutation, the corresponding mask is $Pm(\theta)$. If an intervention operator I_m acts on the hidden vector, then the transported comparison is

$$I_{Pm}(Ph) = PI_m(h).$$

For a fixed mask, the intervention therefore commutes with the permutation after the output is placed in the same coordinate frame. If each representative independently recomputes its mask, the two masks need not satisfy this relation; the resulting intervention outputs can then differ even though the dense function is unchanged.

Procedure-level output. A model-producing procedure has a different output type. Let $\mathcal{P}(\theta)$ be a sparse or otherwise modified model, and let $\widehat{g}$ transport a model from the coordinates of θ to those of $g \cdot \theta$. The appropriate comparison has two parts:

$$\mathcal{P}(g \cdot \theta) \quad \text{versus} \quad \widehat{g}\big(\mathcal{P}(\theta)\big),$$

and, at the resulting-function interface,

$$f_{\mathcal{P}(g \cdot \theta)} \quad \text{versus} \quad f_{\widehat{g}(\mathcal{P}(\theta))}.$$

The first comparison tests whether the procedure returns corresponding structures. The second tests whether those structures implement the same resulting function. These questions are related but not identical: two sparse parameter arrays may differ while realizing the same sparse function, or may have similar masks while producing different outputs because of changed surviving weights.

Why raw equality is not a universal rule. The examples show why one equality test cannot serve every output. A scalar is compared by equality, a labeled vector by transport, a structural object by its induced correspondence, and a new model at the resulting-function interface. Output Semantics records this choice before the paired outputs are inspected.

C.3 Exact-Functional Well-Definedness Audit

The audit is a finite experimental procedure for testing a claim against a declared exact transformation family. It does not replace the definition of the complete fiber.

Step 1: certify the dense pair. Start with θ and a transformation g for which an analytic identity establishes

$$R_A(g \cdot \theta) = R_A(\theta).$$

For a permutation, this follows by reindexing the hidden sum. For a reciprocal gauge, it follows by cancellation:

$$(W_d D^{-1})(Du) = W_d u.$$

A numerical residual on a finite batch is recorded as an implementation check, not as the proof of equality on all X.

Step 2: declare the output semantics. Before running the analysis, specify whether the output is an invariant scalar, a labeled object, a transported structure, or a model-producing procedure. Also specify the comparison map τ_g or the model transport $\widehat{g}$. This prevents a raw array mismatch from being interpreted before its scientific meaning has been fixed.

Step 3: apply the same analysis. Run the same analysis with the same implementation choices on θ and $g \cdot \theta$. When the question concerns independent selection, allow each representative to compute its own output. When the question is a transported control, hold the scientific object fixed and transport it explicitly.

Step 4: compare at the declared level. Evaluate

$$H(g \cdot \theta) \quad \text{against} \quad \tau_g(H(\theta)),$$

or compare the two realized output functions when H returns a model. Report raw disagreement as a diagnostic, but use failure after the prescribed correspondence as the solution-level inconsistency.

Step 5: state the scope of the verdict. A positive result should identify the tested transformation family, output semantics, interface, numerical tolerance, and sampled representatives. It should be described as consistency on that tested family unless the complete fiber has been characterized analytically.

Why one failure is sufficient. A universal solution-level claim has the form

$$\forall \eta \in M_{f_\theta}, \qquad H(\eta) \text{ agrees with } H(\theta) \text{ under the declared semantics.}$$

One certified $\eta \in M_{f_\theta}$ for which the required relation fails is a counterexample to this universal statement. The conclusion does not require showing that every other representative fails.

Why several passes are not a proof over the full fiber. Let G be a known exact transformation group. The tested orbit satisfies

$$G \cdot \theta \subseteq M_{f_\theta},$$

but this inclusion may be strict. Agreement on $G \cdot \theta$ proves consistency on that orbit only. A representation outside the known orbit may still produce a failure, including one arising from a different continuous branch, a discrete branch, an inactive channel, or a singular factorization. Thus a finite empirical pass supports a bounded claim, while a full-fiber theorem requires a separate characterization of M_{f_θ}.

A minimal audit report. For reproducibility, an audit report should state:

1. the architecture, parameterization, input domain, and output interface;
2. the analytic identity defining the exact transformation;
3. the numerical residual, dtype, device, and tolerance used for implementation checking;
4. the analysis and all randomness or calibration choices;
5. the Output Semantics and transport map used for comparison;
6. the raw and semantics-aware outputs, together with the scope of the verdict.

C.4 From Representation Dependence to Consistency Repair

Consistency repair uses the known transformation law either to construct a better observable or to align representatives before applying a coordinate-dependent procedure.

Repair by composition. Suppose two positive representation-local quantities transform as

$$a_i' = d_i a_i, \qquad b_i' = d_i^{-1} b_i, \qquad d_i > 0.$$

Their product is invariant:

$$a_i' b_i' = (d_i a_i)(d_i^{-1} b_i) = a_i b_i.$$

For example, an activation magnitude can be paired with a downstream-column norm. Either factor changes under the reciprocal gauge, but their composition does not. A selector built from the product can therefore be compared as an invariant score, subject to ties, zero entries, and finite precision.

Repair by balancing the representative. Let p_i denote a value-side channel vector and q_i the corresponding downstream vector. For nonzero norms, choose

$$d_i^\star = \sqrt{\frac{\|q_i\|_2}{\|p_i\|_2}}, \qquad p_i^\star = d_i^\star p_i, \qquad q_i^\star = (d_i^\star)^{-1} q_i.$$

The transformed norms are equal:

$$\|p_i^\star\|_2 = \sqrt{\frac{\|q_i\|_2}{\|p_i\|_2}}\,\|p_i\|_2 = \sqrt{\|p_i\|_2\|q_i\|_2},$$

$$\|q_i^\star\|_2 = \sqrt{\frac{\|p_i\|_2}{\|q_i\|_2}}\,\|q_i\|_2 = \sqrt{\|p_i\|_2\|q_i\|_2}.$$

For example, if $\|p_i\|_2 = 4$ and $\|q_i\|_2 = 16$, then $d_i^\star = 2$, giving transformed norms 8 and 8. The construction chooses a balanced representative within the declared positive-diagonal gauge family; it does not claim to identify a unique point in the complete exact fiber.

Repair pipeline. For a procedure $\mathcal{P}$, let C be the balancing map and T_θ the transport from the balanced output back to the original coordinates. The full construction is

$$\theta \longrightarrow C(\theta) \longrightarrow \mathcal{P}(C(\theta)) \longrightarrow T_\theta\big(\mathcal{P}(C(\theta))\big).$$

The procedure is unchanged; only its input frame is controlled. The final output is then compared with the corresponding output obtained from the other representative.

What the repair does and does not establish. Composition establishes invariance under the specified group. For nonzero paired channel norms, balancing selects exactly the same labeled representative throughout a positive-diagonal orbit. Near-zero norms can cause numerical conditioning problems; zero norms are outside this regular-domain definition. Ties concern the subsequent selector and its correspondence, rather than uniqueness of the balancing factors. Additional fiber redundancies require a separate analysis. Improved mask or weight agreement alone does not establish task utility.

C.5 Transport rules for meaningful comparisons

A correspondence must express the scientific object being compared, independently of the observed result. A permutation sends a channel score or mask to the corresponding channel labels; a positive diagonal gauge sends a down-projection weight to WD^{-1} and leaves the support labels unchanged. These maps obey identity and composition. Defining an arbitrary output-dependent map merely to match two observed answers is not an audit.

If $h\theta = \theta$ is a stabilizer, equivariance requires $H(\theta) = \tau_h H(\theta)$. For two group elements reaching the same representative, this stabilizer condition prevents path-dependent reported objects. A unique labeled top-one choice among indistinguishable channels may fail it; a tied set or another explicitly declared object can be appropriate. Equation 14 applies to invariant outputs in a common codomain. Equivariant coordinate arrays can instead represent one transported object on a known orbit; their equivariance is not a characterization of the full functional fiber.

For example, the identity map $H(\theta) = \theta$ simply returns the model parameters. If the output uses the same transformation, $\tau_g = g$, then $H(g\theta) = g\theta = \tau_g H(\theta)$: both sides return the same transformed parameter vector. Thus equivariance holds automatically, without establishing that individual weights are properties of the realized function. For model-producing procedures, the primary scientific comparison is the resulting function, with transported masks and weights serving as additional diagnostics. One unequal output on a declared input refutes all-input equality; agreement on a finite batch cannot prove it.

Exactness of the balanced slice. The balancing factors above obey $c_i(g_D\theta) = c_i(\theta)/d_i$ whenever both norms are nonzero. The balanced weights therefore coincide exactly across this orbit. Fixed, corresponding tie handling on identical balanced states preserves this property. Ordinary floating-point evaluation can perturb both the balanced states and the reconstruction; the controlled measurements in Appendix E.4 distinguish these errors. Singular channels and transformations outside the declared group require separate analysis.

Orbit-invariant products need not identify a full-fiber quantity. For the fixed two-channel linear architecture $f(x) = (v_1u_1+v_2u_2)x$, set $u = (1, 1)$ and compare $v = (1, 1)$ with $v = (1/2, 3/2)$. Both realize $2x$ for every real input. The individual channel products are invariant under reciprocal positive scaling, but differ between these members of the same functional fiber. Thus the composition repair proves the declared orbit-level statement without automatically proving a full-fiber claim.

D Experiments: Auditing Analyses on the Same Exact Solution

This appendix expands Section 4 with worked examples. The examples follow the same order as the main experimental argument: first certify an exact-equivalent pair, then apply an analysis or procedure, then compare its output under the declared Output Semantics. Numerical values below are reported to illustrate the comparison; the full experimental matrices and manifests remain the source of the aggregate results.

D.1 Transformer Components and Experimental Protocol

Transformer language models provide a useful testbed because their modular attention and MLP blocks expose several internal interfaces within one architecture. Coordinate permutations, reciprocal rescalings, and value/output transformations can therefore be written as explicit function-preserving operations and traced from internal representations to selectors, pruning procedures, and downstream outputs. Each certificate is stated for its corresponding module and interface; the criterion itself is not specific to Transformers.

Component–transformation map. The operation acts on an internal coordinate or factorization; the certificate states what is preserved; the audit then compares the output at its declared interface.

Component	Representative operation	Preserved object	Audited output
MLP/SwiGLU	Channel permutation or $u' = Du, W'_d = W_dD^{-1}$	Dense MLP contribution	Selector, intervention
Attention	Head/block relabeling or $v' = Pv, W'_O = W_OP^{-1}$	Declared attention output	Head/value structure
Factorization	$U' = UG^{-1}, V' = GV$	Linear product UV	Weights, masks
Normalization	Gain rescaling with inverse consumer	Only the declared module interface	Hidden-state output
Procedure	Apply pruning to exact-equivalent inputs	Not a gauge identity; compare resulting functions	Sparse model, token, task

MLP/SwiGLU channels. Let h be the input to an MLP block and let the channel contribution before the downstream projection be

$$u_i(h) = \mathrm{SiLU}((W_{\mathrm{gate}}h)_i)\,(W_{\mathrm{up}}h)_i.$$

For a positive diagonal $D = \mathrm{diag}(d_1, \ldots, d_m)$, the value-side contribution and downstream projection transform as

$$u'(h) = Du(h), \qquad W'_{\mathrm{down}} = W_{\mathrm{down}}D^{-1}.$$

Consequently, $W'_{\mathrm{down}}u'(h) = W_{\mathrm{down}}u(h)$ for every h. The dense function is preserved, although activation-only scores and downstream-weight scores change by reciprocal factors. This is the Qwen construction used by the selector and intervention experiments below.

Attention heads and value/output coordinates. Write v for the concatenated value-side output of a declared attention module and W_O for its output projection. A block permutation P gives

$$v' = Pv, \qquad W'_O = W_OP^{-1}, \qquad W'_Ov' = W_Ov.$$

The same local identity covers a relabeling of attention heads when P permutes complete head blocks. A diagonal value/output rescaling has the same form with P replaced by an invertible diagonal matrix. In a full Transformer, positional encodings, grouped-query ordering, tied parameters, and all downstream consumers must be included in the certificate; the module identity alone does not certify the full model unless those conditions are verified.

Factorization and normalization interfaces. For any invertible G, a linear factorization satisfies

$$U'V' = (UG^{-1})(GV) = UV.$$

This algebra also appears in parameterized neural blocks, but an apparently similar rescaling is not automatically valid through a normalization layer. The normalization rule, affine gains, numerical stabilizer, and downstream consumer must be included in the declared interface. These conditions explain why the appendix records architecture-specific certificates rather than claiming that every Transformer rescaling preserves the complete model.

From components to audit outputs. The component operation and the analysis output are separate parts of the audit. A hidden-channel ranking is compared after the induced relabeling or transport; a selector is checked together with the intervention it defines; and a pruning procedure is compared through the function realized by its sparse output. Thus the same internal transformation can be harmless for one output interface and consequential for another.

Implementation-level parameter maps. The GPT-2 construction uses the layer-0 tensors `block.mlp.c_fc.weight`, `block.mlp.c_fc.bias`, `block.mlp.c_proj.weight`, and `block.mlp.c_proj.bias`. Hugging Face Conv1D stores weights with shape $[\text{in}, \text{out}]$, so a hidden-channel permutation p selects columns of the upstream tensor and rows of the downstream tensor:

$$W'_{\text{fc}}[:, j] = W_{\text{fc}}[:, p_j], \quad b'_{\text{fc},j} = b_{\text{fc},p_j}, \quad W'_{\text{proj}}[j, :] = W_{\text{proj}}[p_j, :].$$

If z is the post-activation hidden row vector, then $z'_j = z_{p_j}$ and

$$z'W'_{\text{proj}} = \sum_j z_{p_j} W_{\text{proj}}[p_j, :] = zW_{\text{proj}}.$$

The projection bias is unchanged, and all parameters outside this MLP are copied unchanged.

The Qwen2.5 construction acts on `mlp.up_proj.weight` and `mlp.down_proj.weight`; these checkpoints have no corresponding projection biases. The implemented parameter map is

$$W'_{\text{up}} = DW_{\text{up}}, \qquad W'_{\text{down}} = W_{\text{down}}D^{-1}, \qquad W'_{\text{gate}} = W_{\text{gate}}.$$

With the gate branch fixed,

$$u(h) = \text{SiLU}(W_{\text{gate}}h) \odot (W_{\text{up}}h), \qquad u'(h) = \text{SiLU}(W_{\text{gate}}h) \odot (DW_{\text{up}}h) = Du(h).$$

Therefore $W'_{\text{down}}u'(h) = W_{\text{down}}u(h)$ for every h. All other parameters are unchanged; because the MLP output is identical for every hidden input h, the residual stream and all subsequent computations are identical, hence the complete input-to-logit function is preserved. The finite-batch checkpoint residuals in Table 4 check this implementation in floating-point arithmetic; they are not the exactness certificate. For general non-dyadic gauges, stored FP32 tensors approximate the real-arithmetic map. Appendix E.3 separately verifies an exactly representable stored-weight transformation.

A reciprocal gauge in three channels. A small vector example makes the dense-preserving construction explicit. Let

$$u = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \qquad W_d = \begin{bmatrix} 2 & 1 & 3 \end{bmatrix}, \qquad D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1/4 \end{bmatrix}.$$

The transformed quantities are

$$u' = Du = \begin{bmatrix} 1 \\ 8 \\ 0.75 \end{bmatrix}, \qquad W'_d = W_dD^{-1} = \begin{bmatrix} 2 & 1/4 & 12 \end{bmatrix}.$$

Although both the internal vector and the downstream weights change,

$$W_du = 13 \qquad \text{and} \qquad W'_du' = 13.$$

The two parameterizations therefore have different internal coordinates but the same consumed value. This is the local identity lifted through the Qwen SwiGLU construction.

Table 4: Checkpoint-level numerical exactness checks for the transformations used in the main audit. For Qwen2.5-7B, the residual aggregates the audited layer-0/1 extension rows. Values are maxima over the recorded finite-batch rows; they validate implementation and finite-precision behavior and do not replace the analytic identities.

Model	Layer module	Transformation	dtype	max abs. logit residual	relative residual
GPT-2	0 / MLP	hidden-channel permutation	FP32	1.14×10^{-4}	1.85×10^{-7}
Qwen2.5-0.5B	0 / SwiGLU	positive reciprocal gauge	FP32	7.58×10^{-5}	1.41×10^{-6}
Qwen2.5-1.5B	0 / SwiGLU	positive reciprocal gauge	FP32	7.41×10^{-4}	4.66×10^{-6}
Qwen2.5-7B	0–1 / SwiGLU	positive reciprocal gauge	FP32	3.33×10^{-4}	5.77×10^{-6}

Why exactness is checked before analysis. If the compensation is omitted, the transformed output becomes

$$W_d u' = 2 + 8 + 2.25 = 12.25,$$

which is not the original value 13. A subsequent selector or pruning difference would then mix representation dependence with a changed dense function. The experimental order removes this ambiguity: the compensated pair is the object under audit, while the uncompensated construction is a function-changing control.

Output comparison in the same example. The activation magnitudes of the original and transformed vectors are

$$|u| = (1, 2, 3), \qquad |u'| = (1, 8, 0.75).$$

An activation-only top-two selector chooses channels $\{2, 3\}$ in the first representation and $\{1, 2\}$ in the second. This raw difference is meaningful as a representation-local selector change, but it is not yet a solution-level failure: a transported selector or a composed score must be checked next.

D.2 A Positive Control: Label Transport

Two-channel ranking. Let a ranking analysis return

$$r(\theta) = \begin{bmatrix} 0.9 \\ 0.2 \end{bmatrix}$$

for two hidden channels. After swapping the channels,

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad r(P \cdot \theta) = Pr(\theta) = \begin{bmatrix} 0.2 \\ 0.9 \end{bmatrix}.$$

The raw vectors differ and their top-one index changes from 1 to 2. Inverse transport gives

$$P^{-1} r(P \cdot \theta) = \begin{bmatrix} 0.9 \\ 0.2 \end{bmatrix} = r(\theta).$$

Thus a raw ranking mismatch can be entirely due to a changed coordinate frame.

The GPT-2 positive control is the model-scale version of this example. The hidden-coordinate permutation preserves the dense computation; the raw top-k Jaccard is 0, whereas inverse transport restores a Jaccard of 1.0. The control therefore validates the comparison rule for labeled outputs rather than claiming that all parameter-space differences are harmless.

Transported mask. For a mask $m = (1, 0)^\mathsf{T}$, the corresponding mask after the swap is $Pm = (0, 1)^\mathsf{T}$. Comparing m directly with Pm reports disagreement, while comparing $P^{-1}Pm$ with m reports exact agreement. The same distinction applies to hidden-channel interventions: the mask must be transported before deciding whether the intervention structure changed.

D.3 Selector and Intervention: A Direct Consequence

The three-channel example above separates a changed local selector from the intervention result. Let M be a diagonal mask. When the same scientific mask is transported consistently, diagonal matrices commute:

$$W'_d M u' = W_d D^{-1} M D u = W_d M u.$$

For example, if $M = \mathrm{diag}(1, 0, 1)$, both parameterizations remove channel 2 in their own coordinates and produce the same consumed output after the correspondence is respected.

Independent reselection is different. In the example, the activation top-two masks are

$$M_{\mathrm{act}} = \mathrm{diag}(0, 1, 1), \qquad M'_{\mathrm{act}} = \mathrm{diag}(1, 1, 0).$$

They are not the transported versions of one another. Applying these two independently selected masks gives

$$W_d M_{\mathrm{act}} u = 11, \qquad W'_d M'_{\mathrm{act}} u' = 4.$$

The dense values are equal before intervention, but the independent local selectors lead to different intervention outputs. This is the logic tested by the Qwen selector-intervention matrix.

The model-scale matrix contains 1,440 rows across two checkpoints, three layers, four gauge conditions, five gauge directions, four removal fractions, and three selectors. The fixed-mask control remains at a median gap of 7.12×10^{-7} on non-identity gauges, while the median intervention gaps are 0.176 for the activation selector and 0.530 for the downstream-weight selector. The composed selector has mask Jaccard 1.0 and zero recorded intervention gap. The toy calculation explains why the fixed-mask and independent-selector conditions answer different questions.

D.4 Procedure Dependence

Why Wanda is a useful comparison. Consider one output row with

$$|W| = (2, 1), \qquad \|X_{:,1}\|_2 = 3, \qquad \|X_{:,2}\|_2 = 2.$$

The Wanda scores are $(6, 2)$. Under a reciprocal gauge with $d = (4, 1/2)$,

$$|W'| = (1/2, 2), \qquad \|X'_{:,1}\|_2 = 12, \qquad \|X'_{:,2}\|_2 = 1,$$

so the scores remain $(6, 2)$. The individual factors change, but their product and ordering do not.

Magnitude-only ranking does not have this cancellation: its scores change from $(2, 1)$ to $(1/2, 2)$, reversing the ordering in this toy example. SparseGPT uses a larger reconstruction problem rather than this scalar product, so its behavior must be measured empirically. The model-scale procedure spectrum reflects this difference: Wanda is near-invariant, while Magnitude and SparseGPT show larger discrepancies; balancing substantially reduces the SparseGPT discrepancy.

Mask and full-matrix weight outputs. A mask comparison and a weight comparison answer different questions. If two methods retain the same entries but assign different surviving values, mask Jaccard can be 1 while Δ_W is nonzero. Conversely, two masks can differ while the resulting functions remain close on a particular input set. The transported sparse control applies the exact coordinate map to the base sparse model, keeping its pruning decision fixed. We compare it with independently pruned transformed models using mask Jaccard J and full-matrix weight discrepancy Δ_W.

For the two Qwen checkpoints, the method means are summarized in Table 5.

The table is a procedure spectrum, not a single verdict about pruning. It shows that exact equivalence does not force all procedures to agree, and that a procedure can be made more consistent by changing the representative on which it operates.

D.5 From Sparse Outcomes to Token and Task Behavior

A token-level toy comparison. Suppose the dense model assigns the same next-token distribution to both representatives,

$$p_{\mathrm{dense}} = p'_{\mathrm{dense}} = (0.60, 0.30, 0.10).$$

Table 5: Local procedure comparison: mask Jaccard against the transported sparse control J and full-matrix weight discrepancy Δ_W, averaged over 27 rows per method/checkpoint in the original base frame.

Procedure	Qwen 0.5B (J, Δ_W)	Qwen 1.5B (J, Δ_W)
Magnitude	$(0.835,\ 0.201)$	$(0.841,\ 0.205)$
Wanda	$(0.9999998,\ 8.78 \times 10^{-5})$	$(0.99999994,\ 5.45 \times 10^{-5})$
SparseGPT	$(0.834,\ 0.273)$	$(0.830,\ 0.291)$
Balanced SparseGPT	$(0.988,\ 0.072)$	$(0.978,\ 0.105)$

After independent pruning, let the distributions become

$$p_{\text{sparse}} = (0.70, 0.20, 0.10), \qquad p'_{\text{sparse}} = (0.55, 0.35, 0.10).$$

The top-1 token remains the first coordinate in this pair, although the distributions differ. If instead $p'_{\text{sparse}} = (0.20, 0.70, 0.10)$, top-1 disagreement also appears. This illustrates why both a distributional metric such as Jensen–Shannon divergence and a discrete top-1 metric are useful.

The recorded token comparison follows the same logic. Dense top-1 disagreement is 0 with mean Jensen–Shannon divergence below 8×10^{-12}. Independently pruned pairs show top-1 disagreement 0.051887 and 0.031840 for the two checkpoints, with mean Jensen–Shannon divergences 0.004489 and 0.001997.

Why paired task disagreement is separate. For two correctness vectors on the same problems,

$$c = (1, 1, 0, 0), \qquad c' = (1, 0, 1, 0),$$

both models have accuracy $1/2$, but their correctness disagreement is

$$D_{\text{corr}} = \frac{1}{4}\left(\mathbf{1}[1 \neq 1] + \mathbf{1}[1 \neq 0] + \mathbf{1}[0 \neq 1] + \mathbf{1}[0 \neq 0]\right) = \frac{1}{2}.$$

Thus aggregate accuracy can conceal different problem-level behavior. In the paired GSM8K evaluation, the explicit-final-answer evaluation gives pooled median strict correctness disagreement 0.0883, which falls to 0.0299 after repair (Appendix D.11). Sparse-NLL gaps are reported separately because token-level distributional agreement and task-level correctness are different interfaces.

D.6 Calibration and Implementation Choices

Calibration size. The calibration-size study uses sizes $4, 16, 32, 64, 128, 512$ with historical maximum sequence length 64. For Qwen2.5-0.5B, the independent output-gap median changes from 0.1121 at size 4 to 0.0453 at size 512, while the repaired gap changes from 0.0139 to 0.0044. For Qwen2.5-1.5B, the corresponding independent values are 0.1320 and 0.0670, and the repaired values are 0.0283 and 0.0142. The effect size depends on calibration, but the independent gap remains above the repaired gap across the reported endpoints.

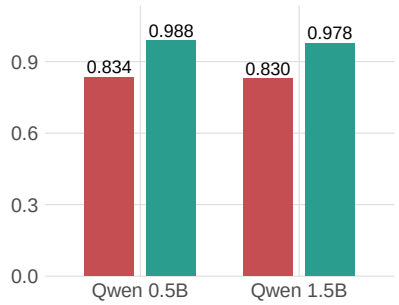
Capped-text allocations and effective budgets. The following products are equal upper bounds, rather than equal observed token budgets, because the original experiment truncated separately sampled texts without packing:

$$128 \times 2048 = 256 \times 1024 = 512 \times 512 = 262{,}144.$$

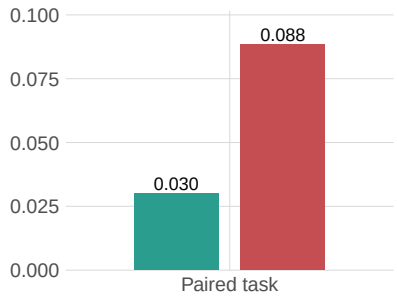
The mean mask Jaccard against the transported control for the 0.5B checkpoint changes from 0.896 to 0.926 to 0.940 across these allocations; for the 1.5B checkpoint it changes from 0.889 to 0.926 to 0.943. The corresponding effective-token medians are 17,921, 36,548, and 73,128; sample count, token count, and context length therefore vary together. A separate control uses exactly 262,144 tokens from one stream and gives Jaccards 0.9471–0.9483 for 0.5B and 0.9531–0.9546 for 1.5B across these allocations (Appendix D.10).

Implementation sensitivity. The implementation-sensitivity grid varies damping, block size, activation order, pruning order, and error compensation while holding final density fixed. The stored traces show that these choices alter intermediate reconstruction statistics. Because continuous intermediate mask distances were not stored, the evidence supports implementation sensitivity but does not identify one unique causal stage. All 384 nominal zero-damping conditions used a nonzero stabilizer, so they do not establish damping-free failure. The separate covariance and precision controls in Appendix E distinguish scalar-damping dependence from reconstruction roundoff.

D.7 Consistency Repair



(a) Mask Jaccard (red: SparseGPT; green: balanced repair).



(b) GSM8K strict correctness disagreement (red: independent sparse pair; green: balanced repair).

Figure 4: Consistency repair reduces structural and behavioral disagreement. Balanced representative construction raises mask agreement for both Qwen checkpoints and lowers paired GSM8K strict correctness disagreement.

The local repair comparison gives activation Jaccard $0.2995 \to 1.0$. Independent balancing implementations reach mask Jaccards 0.9875 and 0.9884. The procedure grid and paired task runs below use separate declared protocols; these measurements are retained alongside the full diagnostic plots.

Composed observable. In the three-channel example, pair the activation magnitude with the corresponding downstream magnitude:

$$a = (1, 2, 3), \qquad b = (2, 1, 3), \qquad a' = (1, 8, 0.75), \qquad b' = (2, 1/4, 12).$$

The coordinate-wise products are

$$a \odot b = (2, 2, 9) = a' \odot b'.$$

The activation-only ranking changes, but the composed observable is unchanged. This is the concrete mechanism behind the composed selector: reciprocal factors cancel before ranking.

Balanced representative. If a value-side channel has norm $\|p_i\|_2 = 4$ and its downstream partner has norm $\|q_i\|_2 = 16$, the balancing factor is

$$d_i^\star = \sqrt{16/4} = 2.$$

The transformed norms are

$$\|d_i^\star p_i\|_2 = 8, \qquad \|(d_i^\star)^{-1} q_i\|_2 = 8.$$

Thus the representative is moved to a balanced frame without changing the dense function. A coordinate-dependent procedure is then applied in this common frame and its output is transported back.

The recorded repair results follow this construction. For standard versus balanced SparseGPT, mask Jaccard against the transported sparse control increases from $(0.834, 0.830)$ to $(0.988, 0.978)$ across the two checkpoints. In the paired task runs, mask Jaccard increases from $(0.788, 0.774)$ to $(0.989, 0.982)$, sparse-NLL gaps decrease from $(0.00369, 0.00334)$ to $(0.000565, 0.000262)$, and

the explicit-final-answer evaluation gives pooled median strict correctness disagreement 0.0883 before repair and 0.0299 after repair. The robustness across gauge strengths and target sparsities is shown in Figure 5. These separate standard and repair matrices use different gauge seeds; their aggregated curves are descriptive comparisons rather than matched before–after effects.

Absolute utility is reported separately for the explicit-final-answer evaluation in Table 9. Accuracy, answer coverage, cap rates, and matched held-out WikiText NLL are shown together; improved pairwise consistency alone does not establish improved task utility or dense-performance recovery.

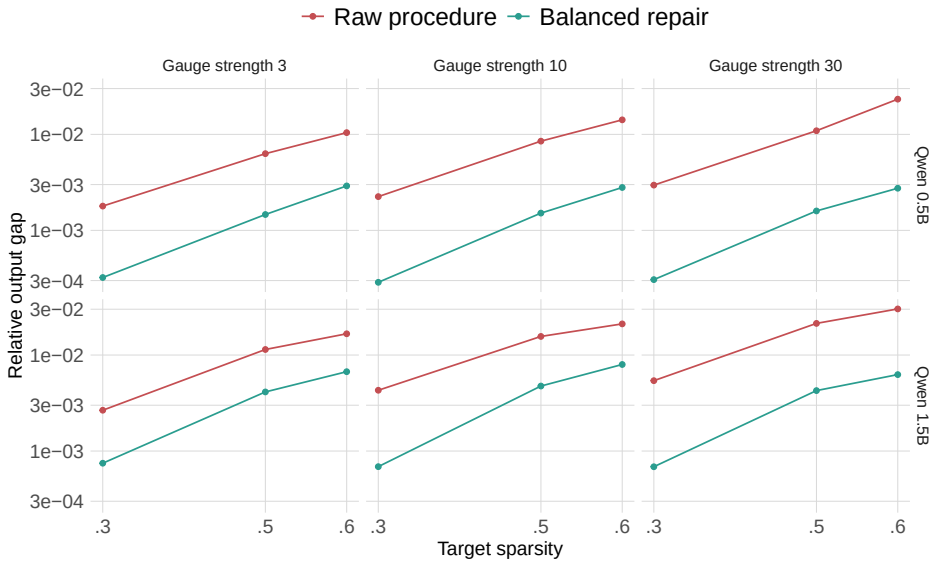


Figure 5: Repair robustness across gauge strengths and target sparsities. The detailed small-multiple view complements the headline before–after comparison in Figure 4.

Limits illustrated by simple edge cases. If $\|p_i\|_2 = 0$, the balancing factor is undefined. If two channels have equal scores, the selector needs a corresponding tie rule; this is separate from uniqueness of nonzero norm-balancing factors. If a channel is near zero, finite precision can make the computed factor unstable. These edge cases explain why the construction is a repair for the declared positive-diagonal gauge family, not a canonicalizer for the complete exact fiber.

D.8 Additional Scope Checks

A numerical implementation check applies a diagonal value/output gauge to a TinyLlama attention module with the correct grouped-query ordering. On the declared finite batch, the dense relative gap is 8.62×10^{-7}; this checks the stated module identity on that batch and does not establish a general cross-architecture pruning claim.

Two interface examples clarify why the output must be declared. Adding the same constant to every logit changes the logit array but leaves softmax probabilities unchanged:

$$\mathrm{softmax}(z + c\mathbf{1}) = \mathrm{softmax}(z).$$

By contrast, fake quantization can produce different deployment errors for exact-equivalent representations because quantization is applied to representation-dependent coordinates. These examples separate equality at the logit, probability, decoded-token, and deployment interfaces.

The residual-fiber check provides the complementary geometric warning: a known orbit can pass transport while a representation outside that orbit remains a distinct exact-fiber witness. Agreement on a tested orbit is consequently a bounded audit result, not a full-fiber theorem.

Table 6: Traceable scale coverage for the held-out utility extension. Each entry in the run column gives the standard/balanced base–transformed comparison record counts; raw and repaired mask scores use the transported sparse controls and balanced-repair outputs, respectively.

Checkpoint	Records	Transformation family	Audited output	Raw mask J	Repair mask J
0.5B	180/180	positive diagonal SwiGLU gauge	held-out mask/NLL	0.743	0.999
1.5B	180/180	positive diagonal SwiGLU gauge	held-out mask/NLL	0.738	1.000
7B	180/180	positive diagonal SwiGLU gauge	held-out mask/NLL	0.802	1.000

D.9 Scale Coverage

The core selector and procedure results use Qwen2.5-0.5B and Qwen2.5-1.5B. The same positive-diagonal SwiGLU reciprocal gauge was also evaluated in separate held-out utility matrices for Qwen2.5-7B. The 7B rows extend the scale coverage. Each method has 540 base–transformed comparison records across the three checkpoints. Together the two methods have 1,080 comparison records and 2,160 individual base/transformed reconstructions; their gauge seeds differ. In the table, the raw and repaired mask scores are medians over the corresponding matrix rows, respectively.

D.10 Executable protocol, comparison frame, and statistical units

The protocol below specifies the experiments behind the existing main figures and procedure table. Core procedure and task sparsity refers to entries of one layer-0 down-projection matrix, while the selector grid varies MLP layers and whole-channel removal fractions. The functional criterion and component certificates are not restricted to that experimental choice.

Common comparison frame. Positive diagonal scaling leaves coordinate labels and zero patterns unchanged, but it changes nonzero weight values. For base sparse weights W_s and independently pruned lifted weights W'_s, the original-base-frame comparison is

$$\Delta_W = \frac{\|W_s - W'_s D\|_F}{\max(\|W_s\|_F, 10^{-30})}, \qquad J = \frac{|\operatorname{supp}(W_s) \cap \operatorname{supp}(W'_s)|}{|\operatorname{supp}(W_s) \cup \operatorname{supp}(W'_s)|}.$$

We set $J = 1$ if the union is empty. This is a full-matrix discrepancy, not a norm restricted to the intersection of surviving entries. It remains a diagnostic in the declared reference frame rather than a coordinate-invariant distance on EFSS; resulting-function comparisons carry the function-level interpretation. For balanced base/lifted reconstructions $\widetilde{W}_0, \widetilde{W}_1$ and their balancing diagonals B_0, B_1, the two matrices compared in the original frame are $\widetilde{W}_0 B_0$ and $\widetilde{W}_1 B_1 D$. All four procedures use this same frame. The local output gap is $\|Y_0 - Y_1\|_F / \max(\|Y_0\|_F, 10^{-30})$ on the declared common inputs. A finite input set measures a discrepancy; it cannot establish all-input functional equality.

Selector grid. The selector–intervention grid uses Qwen2.5-0.5B layers $\{0, 12, 23\}$ and 1.5B layers $\{0, 14, 27\}$, $\kappa \in \{1, 3, 10, 30\}$, gauge-direction seeds 7101–7105, removed channel fractions $r \in \{0.05, 0.10, 0.20, 0.30\}$, and three selectors. Thus $2 \times 3 \times 4 \times 5 \times 4 \times 3 = 1,440$ recorded comparisons. The retained count is $\operatorname{round}(n(1-r))$; top-$k$ keeps the highest scores. For each batch the activation score is the mean absolute post-gate, pre-down-projection channel activation; these batch means are averaged. The weight score is the downstream column ℓ_1 norm, and the composed score is their product. The calibration comprises 32 WikiText training texts, selection seed 260826, maximum 256 tokens, batch size two. Padding positions participate in these local activation summaries; the fixed-mask and independent-selector comparisons use the same inputs.

The selector–intervention grid and the procedure and token-level controls described below draw a CPU FP64 Gaussian direction z, scale its range to one, multiply by $\log \kappa$, center its mean, and exponentiate before converting to FP32. Hence $\max d_i / \min d_i = \kappa$ up to rounding. The identity condition has $\kappa = 1$. The reported intervention gap compares two selected masks on the same base

local computation, separating selection from transport roundoff. Fixed-mask gaps instead compare corresponding interventions across base and lifted computations. Reported selector–intervention medians exclude the identity gauges; composed-mask Jaccard is one throughout this grid.

Procedure grid. The procedure grid uses 16 WikiText training texts, seed 8404, maximum 256 tokens, batch size two, layer-0 down-projection only, gauge-direction seeds 7401–7403, $\kappa \in \{3, 10, 30\}$, and sparsities $\{0.3, 0.5, 0.6\}$. Both Qwen tokenizers give 2,299 valid tokens in 2,928 padded positions. The grid has 108 rows per checkpoint and 27 per method/checkpoint. Each base reconstruction is reused across its matching gauge draws. Entries in Table 5 are arithmetic means over those 27 rows. They are separate from the task protocol and held-out scale matrices.

Magnitude pruning thresholds the entire matrix by absolute weight with a rounded keep count. Wanda retains a rounded number in each output row, using $|W_{ri}|\sqrt{\sum_t X_{ti}^2}$. SparseGPT uses the vendored unstructured reconstruction, scalar damping $0.01\,\mathrm{mean}(\mathrm{diag}\,H)$, block size 128, natural column order, and sequential error compensation. Its threshold convention can remove one additional entry per block, so realized density is stored in the result records. No quantization or fine-tuning is applied. Balanced SparseGPT independently norm-balances each dense representative, uses the same reconstruction rule, and is transported back as defined above. The historical padded calibration is retained in this grid to isolate the common-frame weight comparison. Appendix D.10 adds calibration without padding.

Token metrics. The token-level replication uses calibration train seed 7801 (16 texts, maximum 256 tokens, batches of two) and evaluation test seed 8801 (eight texts, maximum 256 tokens, batches of two), layer zero, sparsity 0.5, gauge-direction seed 7802, and $\kappa = 10$. Calibration has 2,619 valid positions out of 3,100; evaluation has 848 out of 1,152. Calibration padding is retained, but both evaluation metrics use only positions with attention mask one, including the final valid input token's next-token distribution. Let V be these positions, and p_t, q_t be the two vocabulary distributions. We report

$$D_{\mathrm{top1}} = |V|^{-1} \sum_{t \in V} \mathbf{1}[\arg\max p_t \neq \arg\max q_t], \quad \mathrm{JS} = \frac{1}{2|V|} \sum_{t \in V} \left[\mathrm{KL}(p_t \| m_t) + \mathrm{KL}(q_t \| m_t)\right],$$

where $m_t = (p_t + q_t)/2$ and logarithms are natural. Evaluation uses FP64 log-softmax and log-add-exp for stable divergence accumulation. Dense and transported sparse controls have $D_{\mathrm{top1}} = 0$ and JS below 8×10^{-12}. Independent sparse pairs give $D_{\mathrm{top1}} = 0.051887/0.031840$ and JS $0.004489/0.001997$.

Shared task calibration and held-out NLL. Task models use one layer-0 down-projection at sparsity 0.5, calibration seeds 101/202, and gauge seeds 7001–7005. The seeded GPU generator draws each $\log d_i$ uniformly from $[-\frac{1}{2}\log 10, \frac{1}{2}\log 10]$, so 10 bounds the range ratio; this differs from the Gaussian-range directions in the procedure grid above. Calibration uses 128 WikiText training texts capped at 64 tokens, right-padded in batches of four. Seeds 101/202 give 7,276/7,149 valid tokens in 8,192 positions. The 64-token setting reproduces the stored input hashes. Held-out NLL uses 128 WikiText test texts, seeds 920101/920202, cap 64 and batch four, with 7,293/7,131 valid input positions. NLL sums shifted-token negative log likelihood and divides by the number of valid shifted labels; padding labels are excluded. No quantization or fine-tuning is applied.

Replication units. Calibration and gauge draws are nested within two fixed checkpoints; base variants are reused across gauges. The main task has 20 paired conditions, not 20 independently trained models. Figure 3 reports ten-condition medians within each checkpoint, and Figure 4 reports pooled medians. Figure 1(b) retains arithmetic means at each gauge range; pooled non-identity selector medians in the text summarize a different, explicitly stated aggregation. The procedure table reports means of 27 conditions per method and checkpoint. None of these condition counts is a model-population sample size.

Packed-stream control. The packed control shuffles the training-text list with seed 260908, tokenizes without added special tokens, inserts EOS between texts, and takes exactly the first 262,144 tokens. The same ordered token stream is reshaped into 128×2048, 256×1024, and 512×512 contexts. Batch size is one and every attention-mask entry is one. We use layer zero, sparsity 0.5,

scalar damping 0.01, block size 128, and Gaussian-range gauge seed 7501 at $\kappa = 10$. Stream hashes agree across all three allocations. A separate numerical check on a 256-token prefix (two contexts of 128) checks the same gauge on each checkpoint: relative dense-logit gaps are below 1.05×10^{-6} and top-1 disagreement is zero for both Qwen checkpoints and TinyLlama. This finite check supplements the analytic certificate; it does not test every packed position. Boundaries change the available context, so this isolates allocation at fixed token content and count, not identical activation statistics. It is one stream and one direction per checkpoint.

Table 7: Exactly 262,144 valid tokens per allocation from one fixed stream, with one direction at $\kappa = 10$. Weight and local-output comparisons are in the original base frame.

Model	Allocation	J	Δ_W	Output gap
0.5B	128×2048	0.9471	0.1328	0.0449
0.5B	256×1024	0.9482	0.1306	0.0446
0.5B	512×512	0.9483	0.1298	0.0447
1.5B	128×2048	0.9531	0.1300	0.0558
1.5B	256×1024	0.9538	0.1285	0.0557
1.5B	512×512	0.9546	0.1268	0.0556

The added TinyLlama layer-0 SwiGLU test uses the same packed 128×2048 protocol and reciprocal gauge. Standard versus balanced mask Jaccard is $0.949876 \rightarrow 0.999433$. This extends a local procedure check to a second family, with one checkpoint, stream, layer, and gauge; its attention-module numerical check in Appendix D.8 remains a separate experiment.

D.11 Completion-aware GSM8K evaluation

The evaluation uses the two Qwen checkpoints, calibration seeds 101/202, five gauges 7001–7005 per calibration, layer-0 down-projection sparsity 0.5, and the full 1,319-question GSM8K test split. Calibration and NLL evaluation retain the separate 128-text, 64-token-cap, right-padded batch-four WikiText protocols and GPU uniform-log gauges of Appendix D.10, with matching calibration hashes. The evaluation interface was fixed after development on a separate 32-question training subset: fixed eight-shot chain-of-thought demonstrations, greedy decoding with a 4,096-token cap, and an explicit final-answer parser. The demonstrations are pinned to the EleutherAI evaluation harness task configuration at commit `b954108c9baaaa934b4ad842033b31a97ee30816`. No chat template or prompt truncation is used. Batch size is four. Model, calibration, task, prompt, evaluator, and protocol hashes are saved with the results.

Each sequence stops independently at EOS or a next-question/terminal boundary (`Q:`, `</s>`, or `<|im_end|>`). Only text before the first boundary is scored. The parser reads the first explicit `The answer is` followed by an optional currency symbol and a signed integer or decimal, permitting thousands separators and a following unit. Decimal canonicalization preserves magnitude. There is no arbitrary last-number fallback. A response is resolved only if this marker yields a number and generation did not reach the cap. Raw text, scored text, generated token IDs, stop reason, lengths, prediction, gold, and resolution flag are retained for every question. The precise regular expression is in the frozen evaluator, and the training pilot's earlier parser annotations are preserved separately.

Strict accuracy treats unresolved responses as failures. We report correctness disagreement and numeric-answer disagreement separately, with the jointly resolved coverage for the latter. Missing answers are not silently excluded from the headline uncertainty. If d jointly resolved questions disagree and u have at least one unresolved response, then the all-question disagreement under arbitrary assignments to unresolved outcomes lies in

$$[L, U] = [d/n, (d + u)/n].$$

These are conservative missing-outcome bounds, not a claim that capped generations would eventually finish. They apply separately to correctness and numeric-answer disagreement. For independent and balanced comparisons, the reduction lies in $[L_{\mathrm{ind}} - U_{\mathrm{bal}}, U_{\mathrm{ind}} - L_{\mathrm{bal}}]$. Shared responses can make these bounds conservative. Condition-wise bounds are aggregated consistently; pooled medians and within-checkpoint means are distinguished.

Table 8: Completion-aware GSM8K. D_{ind} and D_{bal} are strict correctness disagreement D_{corr} for independent and balanced pruning, respectively, reported as medians of ten nested conditions per checkpoint. Reduction and its conditional question-bootstrap interval are means; the last column gives separate conservative missing-outcome bounds on that mean reduction.

Checkpoint	D_{ind}	D_{bal}	Reduction	Bootstrap 95%	Missing bounds
Qwen 0.5B	0.118	0.034	0.082	$[0.073, 0.092]$	$[0.007, 0.155]$
Qwen 1.5B	0.064	0.024	0.040	$[0.033, 0.047]$	$[-0.007, 0.085]$

Table 9: Absolute GSM8K utility and completion, by variant group. Base/scaled denote original/transformed parameterizations. The sparse control transports the base sparse model without pruning again. Accuracy is the mean [minimum, maximum] across the stated number of evaluations; coverage, cap rate, and NLL are means. Shared bases and execution duplicates are not independent fitted models. All percentages use all 1,319 GSM8K questions; NLL uses the matched held-out WikiText batches.

Checkpoint	Variant (n)	Accuracy % [range]	Resolved %	Cap %	NLL
Qwen 0.5B	Dense base (2)	40.9 [40.9,40.9]	95.8	0.7	3.419
Qwen 0.5B	Dense scaled (2)	40.9 [40.9,40.9]	95.8	0.7	3.419
Qwen 0.5B	Sparse base (2)	39.4 [39.1,39.7]	94.6	1.1	3.429
Qwen 0.5B	Sparse scaled (10)	39.8 [38.1,40.9]	94.4	1.2	3.430
Qwen 0.5B	Balanced base (2)	39.5 [39.0,39.9]	94.2	1.0	3.428
Qwen 0.5B	Balanced scaled (10)	39.6 [38.9,40.6]	94.6	1.0	3.428
Qwen 0.5B	Sparse control (2)	39.4 [39.1,39.7]	94.6	1.1	3.429
Qwen 1.5B	Dense base (2)	65.9 [65.9,65.9]	95.1	0.3	3.084
Qwen 1.5B	Dense scaled (2)	65.9 [65.9,65.9]	95.1	0.3	3.084
Qwen 1.5B	Sparse base (2)	66.5 [66.3,66.7]	95.8	0.3	3.091
Qwen 1.5B	Sparse scaled (10)	66.8 [65.4,67.9]	96.1	0.3	3.088
Qwen 1.5B	Balanced base (2)	66.6 [65.7,67.6]	96.5	0.1	3.089
Qwen 1.5B	Balanced scaled (10)	66.8 [66.0,67.7]	96.5	0.1	3.089
Qwen 1.5B	Sparse control (2)	66.5 [66.3,66.7]	95.8	0.3	3.091

Questions are jointly resampled across the ten nested calibration/gauge conditions within each checkpoint for a separate conditional bootstrap (2,000 replicates, seed 260909). Shared base models are not counted as independent fitted models. Sampling intervals and missing-outcome bounds answer different questions and are reported separately. Dense and transported sparse controls are run at gauge 7001 for every calibration; a zero result is checked rather than assumed. Separate 96-token and 384-token evaluations used a last-number fallback and are archived with their own protocols. The present figures use only the explicit-final-answer evaluation; its changed prompt and stopping rule are not interpreted as an isolated parser-only correction.

Results and interpretation. The pooled median strict correctness disagreements are 0.0883 for independent pruning and 0.0299 after balancing. Strict disagreement decreases in all 20 paired conditions. Across the 60 final variant evaluations, resolved-answer coverage is 91.05–97.88%, termination without a parseable final marker is 2.05–7.73%, and cap hits are 0.00–1.90%. These last two unresolved categories are disjoint; no capped response has a parseable marker in this evaluation. Resolution denotes compliance with the stated rule, not a guarantee that all other responses lack a semantically recoverable answer.

Independent pruning differs on jointly resolved final numbers in all 20 conditions, with 126–490 disagreements per condition. Thus the effect is not entirely an artifact of extracting numbers from unfinished reasoning. Mean numeric-answer reduction bounds are $[0.1581, 0.3064]$ for 0.5B and $[0.0293, 0.1215]$ for 1.5B. Correctness treats different wrong answers identically and has the distinct bounds in Table 8. Its mean reduction is not robust to every missing-outcome assignment for 1.5B; 12 of 20 individual conditions have a positive conservative lower bound.

All eight dense and transported-control comparisons have zero strict correctness disagreement and zero jointly resolved numeric-answer differences. There are two raw-text differences. Sixty variant

Table 10: All completion-aware paired conditions. Intervals are conservative all-question bounds for correctness (C) or numeric-answer (A) disagreement. Subscripts ind/bal denote independent/balanced pruning; coverage is the fraction with both final answers resolved in each comparison.

Model	Cal./gauge	C_{ind} bounds	C_{bal} bounds	A_{ind} bounds	A_{bal} bounds	Coverage %
Qwen 0.5B	101/7001	$[0.090, 0.157]$	$[0.032, 0.081]$	$[0.318, 0.385]$	$[0.112, 0.161]$	93.3/95.1
Qwen 0.5B	101/7002	$[0.106, 0.171]$	$[0.042, 0.096]$	$[0.329, 0.394]$	$[0.128, 0.182]$	93.5/94.6
Qwen 0.5B	101/7003	$[0.096, 0.155]$	$[0.030, 0.083]$	$[0.342, 0.400]$	$[0.121, 0.174]$	94.2/94.7
Qwen 0.5B	101/7004	$[0.107, 0.169]$	$[0.033, 0.083]$	$[0.340, 0.403]$	$[0.102, 0.152]$	93.8/95.0
Qwen 0.5B	101/7005	$[0.091, 0.149]$	$[0.031, 0.084]$	$[0.314, 0.371]$	$[0.121, 0.174]$	94.2/94.7
Qwen 0.5B	202/7001	$[0.102, 0.199]$	$[0.027, 0.111]$	$[0.303, 0.400]$	$[0.101, 0.184]$	90.3/91.7
Qwen 0.5B	202/7002	$[0.121, 0.231]$	$[0.039, 0.127]$	$[0.336, 0.447]$	$[0.115, 0.203]$	88.9/91.2
Qwen 0.5B	202/7003	$[0.105, 0.196]$	$[0.027, 0.110]$	$[0.332, 0.424]$	$[0.099, 0.182]$	90.8/91.7
Qwen 0.5B	202/7004	$[0.121, 0.206]$	$[0.021, 0.099]$	$[0.371, 0.457]$	$[0.080, 0.158]$	91.4/92.2
Qwen 0.5B	202/7005	$[0.108, 0.225]$	$[0.026, 0.105]$	$[0.337, 0.453]$	$[0.092, 0.171]$	88.3/92.1
Qwen 1.5B	101/7001	$[0.052, 0.117]$	$[0.019, 0.067]$	$[0.108, 0.174]$	$[0.033, 0.081]$	93.5/95.2
Qwen 1.5B	101/7002	$[0.058, 0.114]$	$[0.023, 0.072]$	$[0.124, 0.180]$	$[0.039, 0.088]$	94.5/95.1
Qwen 1.5B	101/7003	$[0.040, 0.094]$	$[0.027, 0.074]$	$[0.096, 0.149]$	$[0.045, 0.092]$	94.6/95.3
Qwen 1.5B	101/7004	$[0.043, 0.096]$	$[0.023, 0.070]$	$[0.098, 0.151]$	$[0.043, 0.090]$	94.7/95.3
Qwen 1.5B	101/7005	$[0.052, 0.108]$	$[0.022, 0.070]$	$[0.113, 0.168]$	$[0.045, 0.092]$	94.5/95.2
Qwen 1.5B	202/7001	$[0.062, 0.114]$	$[0.025, 0.052]$	$[0.126, 0.178]$	$[0.052, 0.080]$	94.8/97.3
Qwen 1.5B	202/7002	$[0.052, 0.104]$	$[0.017, 0.051]$	$[0.108, 0.160]$	$[0.048, 0.081]$	94.8/96.7
Qwen 1.5B	202/7003	$[0.056, 0.105]$	$[0.018, 0.047]$	$[0.116, 0.165]$	$[0.042, 0.071]$	95.1/97.1
Qwen 1.5B	202/7004	$[0.055, 0.106]$	$[0.015, 0.045]$	$[0.120, 0.171]$	$[0.042, 0.072]$	94.9/97.0
Qwen 1.5B	202/7005	$[0.061, 0.106]$	$[0.024, 0.055]$	$[0.119, 0.165]$	$[0.057, 0.088]$	95.5/96.9

files contain 79,140 continuations; shards and shared bases do not supply additional independent observations. This separates exactness controls, numerical generation sensitivity, and independently reconstructed sparse-model differences.

Repeated-marker sensitivity. Among 79,140 continuations, 38 have multiple parseable markers and one has conflicting values. Taking the last explicit marker changes one resolved prediction and no correctness labels; treating the conflict as unresolved also preserves the aggregate conclusions. These exploratory checks retain the frozen first-marker rule. Full sensitivity records are linked from the supplementary companion's Additional audit details guide.

D.12 Reproduction and execution environment

The anonymous supplementary companion provides model/data manifests, frozen protocols and splits, recorded measurements, scoring and analysis code, and checksums. Its Additional audit details guide links the extended plots, IOI settings, sensitivity records, and execution history. The reproduction guide distinguishes replay of recorded measurements from fresh model execution and specifies coverage; large editing-update tensors are not included in the submission package. Scientific protocols and sampling units remain in Appendices D.10, F.1, and G.

Runs used Python 3.10.12, PyTorch 2.5.1+cu121, Transformers 5.7.0, datasets 4.8.5, NumPy 2.2.6, and pandas 2.3.3 on RTX A6000 GPUs. Ordinary inference and pruning used FP32 with TF32 disabled; FP64 controls are identified separately. The application matrix used six GPUs. Full checkpoint revisions and protocol hashes are retained in the manifests; manuscript tables derive from the recorded analysis files.

E Regularization Covariance and Numerical Reconstruction

This appendix separates three objects: exact equivariance of the local reconstruction algorithm, the regularizer used by an implementation, and finite-precision evaluation. The conditional analysis explains a source of representation dependence in the procedure comparison and the residual after balancing.

E.1 A CONDITIONAL EQUIVARIANCE RESULT FOR SPARSEGPT

Let $W \in \mathbb{R}^{m\times n}$ act on a matrix of calibration activations $X \in \mathbb{R}^{n\times N}$. Write $H = cXX^\mathsf{T}$ for any fixed $c > 0$. The reciprocal positive-diagonal action is

$$W' = WD^{-1}, \qquad X' = DX, \qquad H' = DHD, \qquad D = \operatorname{diag}(d_1, \ldots, d_n) > 0.$$

We consider the unstructured, blockwise SparseGPT reconstruction rule with no quantization, a fixed column order, the same block partition and sparsity, and a deterministic tie rule that selects corresponding coordinates. No activation-dependent reordering, dead-column substitution, or unreported diagonal jitter is included.

Proposition 3 (Regularizer covariance is sufficient). Suppose $A = H + \Gamma(H)$ is positive definite and

$$\Gamma(DHD) = D\Gamma(H)D. \tag{21}$$

Then, in exact arithmetic, the local reconstruction operator $\mathcal{P}_\Gamma$ satisfies

$$\mathcal{P}_\Gamma(WD^{-1}, DX) = \mathcal{P}_\Gamma(W, X)D^{-1}. \tag{22}$$

Consequently, its masks coincide under the diagonal correspondence and its reconstructed local outputs agree for every input transformed by D. This is a sufficient condition for the stated algorithm; its failure does not force disagreement for every individual weight matrix.

Proof. Equation 21 gives $A' = DAD$, hence $(A')^{-1} = D^{-1}A^{-1}D^{-1}$. Let U denote the upper Cholesky factor with $U^\mathsf{T}U = A^{-1}$ and positive diagonal. Uniqueness of this factor gives

$$U' = UD^{-1}, \qquad U'_{ij} = U_{ij}/d_j.$$

At the first block, the saliency score for entry (r, i) is invariant:

$$\frac{(W'_{ri})^2}{(U'_{ii})^2} = \frac{(W_{ri}/d_i)^2}{(U_{ii}/d_i)^2} = \frac{W_{ri}^2}{U_{ii}^2}.$$

Thus thresholding selects the same entries. For a processed column let w_i be its current value, q_i its masked value, and $e_i = (w_i - q_i)/U_{ii}$ its reconstruction error. If the current states correspond, then $q'_i = q_i/d_i$ and $e'_i = e_i$. The update to every remaining column j obeys

$$w'_j - e'_i U'_{ij} = (w_j - e_i U_{ij})/d_j.$$

This preserves the induction hypothesis within the block. The matrix update across blocks is the sum of these same error contributions, so it preserves correspondence as well. The next block therefore has the same scores and selected mask. Induction over all blocks proves the identity. The argument also applies to a fixed structured grouping whose selection rule respects these same corresponding scores. It does not cover a grouping or order recomputed from a non-invariant activation statistic. □

What the standard ridge changes. Undamped reconstruction ($\Gamma = 0$) satisfies the condition whenever $H \succ 0$. A scalar ridge used in the vendored implementation is

$$\Gamma_{\text{scalar}}(H) = \alpha \ \operatorname{mean}(\operatorname{diag} H) I.$$

Its transformed value is generally different from $D\Gamma_{\text{scalar}}(H)D = \alpha \operatorname{mean}(\operatorname{diag} H) D^2$. It therefore introduces a coordinate-dependent quadratic penalty. In contrast,

$$\Gamma_{\text{diag}}(H) = \alpha \operatorname{diag}(H_{11}, \ldots, H_{nn})$$

satisfies Equation 21. For $H \succeq 0$, $\alpha > 0$, and strictly positive diagonal, it also makes $A \succ 0$. This construction is an analytic control for covariance, not a claim of a new state-of-the-art pruning method. A common scaling of H is immaterial for either homogeneous ridge, but nonuniform scaling is different.

An exact scalar-ridge counterexample. The failure is not only a missing sufficient condition. Take one output row and four input coordinates,

$$W = (1, 0.11, 0.01, 0.01), \quad H = \mathrm{diag}(1, 100, 1, 1), \quad D = \mathrm{diag}(10, 1, 1, 1), \quad \alpha = 0.01.$$

Here $H \succ 0$ and can be realized by $X = \mathrm{diag}(1, 10, 1, 1)$ with $c = 1$. Both inverse-curvature factors are diagonal, so reconstruction has no off-diagonal error updates. Before the gauge, the scalar ridge is 0.2575 and the first two scores are

$$W_1^2 A_{11} = 1.2575, \qquad W_2^2 A_{22} = 1.21311575.$$

After the gauge the ridge is 0.505, and these scores become

$$(W_1')^2 A_{11}' = 1.00505, \qquad (W_2')^2 A_{22}' = 1.2161105.$$

The other scores are below 0.001 in either frame. A one-entry retention budget therefore keeps coordinate 1 before the gauge and coordinate 2 afterwards. Transport cannot reconcile the supports; at the base input e_1, the resulting outputs are 1 and 0. These are strict inequalities in exact arithmetic with full-rank curvature, demonstrating a scalar-ridge failure without singularity or roundoff. The covariance result does not require every scalar-ridge instance to fail.

E.2 Independent numerical controls

We captured the layer-0 down-projection inputs of each Qwen checkpoint on 16 WikiText training texts, seed 8404, capped at 256 tokens, batches of two. This is the historical procedure-comparison calibration (2,299 valid positions, 2,928 positions including padding); padding is retained here to isolate the reconstruction change. All comparisons use sparsity 0.5, block size 128, $\alpha = 0.01$, gauge range $\kappa = 10$, and direction seed 7401. The full down-projection matrices are reconstructed, not a small submatrix. We form H from these captured inputs, then compute DHD algebraically. Thus the paired curvature inputs differ only by the declared algebraic map. The FP64 runs promote the captured FP32 inputs before forming H; they do not claim FP64 end-to-end model inference.

Table 11: Controlled reconstruction under algebraically transported curvature. Entries are mask Jaccard and the relative Frobenius weight gap after transport to the base frame. No retry jitter is used.

Arithmetic	Regularizer	Qwen 0.5B (J, Δ_W)	Qwen 1.5B (J, Δ_W)
FP32	scalar	$(0.8165, 0.2964)$	$(0.8212, 0.3075)$
FP32	diagonal	$(0.9800, 0.1029)$	$(0.9674, 0.1399)$
FP64	scalar	$(0.8164, 0.2965)$	$(0.8211, 0.3076)$
FP64	diagonal	$(1.0000, 1.42 \times 10^{-13})$	$(1.0000, 1.12 \times 10^{-13})$

The scalar-ridge discrepancy persists in FP64, while the covariance control reaches identical masks and numerical weight agreement. The FP32 diagonal-ridge discrepancy also shows why an exact theorem alone is insufficient to describe an implementation. Cholesky-based reconstruction and discrete mask selection can amplify small arithmetic errors. We do not infer from this one-direction control that damping is the only contributor in every configuration of the historical implementation sweep.

The local implementation is tested against the vendored SparseGPT algorithm on an identical supplied positive-definite matrix, matching masks and weights. A separate seeded 6×24 weight matrix with 180 calibration samples tests undamped and diagonal-ridge covariance in FP64, as well as a scalar-ridge counterexample. Nonpositive diagonals fail explicitly. The historical implementation-sensitivity sweep labeled some rows "zero damping," but all 384 such detailed rows used a nonzero stabilizer; those rows cannot establish damping-free failure.

E.3 An exactly representable parameter transformation

General diagonal gauges require rounding when stored in FP32. We therefore add a distinct control with $d_i = 2^{k_i}$, where k_i is sampled uniformly from $\{-1, 0, 1\}$ using CPU generator seed 260909. This has range $\kappa = 4$. In both Qwen checkpoints, FP64 checks on the stored tensors verify $W_u' =$

DW_u and $W'_d = W_d D^{-1}$ exactly: these binary-power multiplications/divisions incur no rounding for the observed weights. Thus these serialized parameter pairs satisfy the analytic gauge identity, rather than only approximating the intended parameter map.

We use the layer-0 block, the 16-text seed-8404 calibration of the procedure grid, and the eight-text seed-8801 evaluation of the token check. Native calibration activations satisfy $X' = DX$ bit for bit. Dense logits and transported sparse-control logits are also bit-identical on all 848 valid evaluation positions, for both checkpoints. Standard independent SparseGPT reconstruction nevertheless yields mask Jaccards $0.7397/0.7266$, original-frame weight gaps $0.3692/0.4000$, and top-1 disagreements $0.0625/0.0413$ for 0.5B/1.5B. The independent sparse JS values are $0.00832/0.00252$.

This one-direction control rules out a perturbed dense parameter map or a changed dense output on these probes as the explanation of the observed procedure difference. It does not claim bitwise inference equality on every possible input. The discrete three-level gauge is a separate distribution from the Gaussian-range and uniform-log grids. Complete tensor-map checks, exponent counts, and metrics are retained with the dyadic-control records described in Appendix D.12.

E.4 Why exact balancing can leave a numerical residual

On a regular positive-diagonal orbit let $p_i \neq 0$ be the up-projection row and $q_i \neq 0$ the corresponding down-projection column. With $c_i(\theta) = \sqrt{\|q_i\|_2 / \|p_i\|_2}$, positive scaling gives

$$c_i(g_D\theta) = c_i(\theta)/d_i, \qquad C(g_D\theta) = C(\theta).$$

Thus balancing is an exact, unique norm-balanced representative on this labeled orbit. A deterministic procedure on identical balanced inputs is exactly consistent there. Incompleteness of this orbit in the full fiber cannot explain a nonzero residual within this regular orbit.

We independently balanced the native FP32 base and lifted checkpoints and captured both activation sets. The relative up/down weight gaps are about 6.9×10^{-8} for 0.5B; the activation gap is 1.40×10^{-7}. Reconstructing these independently rounded weights and inputs in FP32 yields mask Jaccard $0.9861/0.9813$ for 0.5B/1.5B. Promoting the same weights and activation sets to FP64 before forming curvature and reconstructing yields $J = 1$ for both, with balanced-frame weight gaps $1.15 \times 10^{-7}/1.27 \times 10^{-7}$. Computing the balancing factors and transformed curvature algebraically in FP64 also yields $J = 1$ and gaps below 10^{-12} for both checkpoints.

These controls locate the observed residual in numerical realization of balancing and reconstruction in this setting, rather than an intrinsic failure of positive-diagonal canonicalization. They do not certify that every gauge, near-zero channel, precision format, or tied selection is numerically stable. Zero channel norms are outside the regular-domain statement; they require a separately specified extension. Arbitrary permutations also need corresponding orders and block partitions, which the fixed-order implementation does not automatically provide.

F A Supporting Test of Component-Importance Interpretation

This supporting audit tests the ordering of actual correct-answer margin losses after selected-channel ablation. It complements the pruning and editing decisions without claiming a unique circuit or a global layer ranking.

F.1 IOI protocol and controls

We adapt the indirect object identification (IOI) templates of Wang et al. (2022) to Qwen2.5-0.5B/1.5B and TinyLlama-1.1B-Chat. Correct and foil names must each extend every complete prompt by one token under all three tokenizers. Development, calibration, and test use disjoint pools of 12/16/18 eligible names and 3/5/7 prompt templates, respectively. Development/calibration/test contain 128/256/1,024 prompts in name-swapped pairs. All checkpoints meet the predeclared 75% candidate-name development accuracy threshold (128/128 each); unrestricted next-token accuracies are 85.2/61.7/80.5%. Inference uses FP32, TF32 off, right padding, batch size 16, and no truncation, demonstrations, chat template, generation, or numeric parser. We evaluate correct-minus-foil logits at the last valid position.

Table 12: IOI layer-order comparisons in joint representations. Entries are point/supported sign reversals; support uses simultaneous 95% bands over all 432 declared contrasts. Denominators count layer pairs, budgets, checkpoints, and gauges, not independent models. Supported reversals are unchanged after deduplicating evaluation pairs.

Selector	Random gauges (54)	Dyadic (18)	Stress (18)
Activation	4/1	2/1	9/9
Composed	0/0	0/0	0/0
Signed Taylor	0/0	0/0	0/0

For depth L, the three candidate layers are $\lfloor L/6 \rfloor$, $\lfloor L/2 \rfloor$, and $\lfloor 5L/6 \rfloor$ (zero-based). At each layer, budgets 5%/10% zero the highest-scored $\mathrm{round}(bd)$ whole channels at every position. Calibration-only scores are mean absolute post-gate activation, that score times the down-column ℓ_2 norm, and signed Taylor influence

$$s_i^{\mathrm{Taylor}} = N^{-1} \sum_{n,t \text{ valid}} a_{nti}\, \partial m_n / \partial a_{nti}.$$

Here m_n is the correct-minus-foil margin. Taylor is a first-order attribution, not a reproduction of Edge Attribution Patching (Syed et al., 2023); effects use actual forward ablations. Ties use ascending channel index. A fixed random ordering per layer supplies nested matched-budget controls. Reciprocal scaling multiplies activation scores by d_i but cancels in composed and Taylor scores; permutations transport all score labels.

Each layer has identity, compensated permutation, three independent uniform-log gauges with $\log d_i \in [-\log(10)/2, \log(10)/2]$, a dyadic gauge $d_i \in \{1/2, 1, 2\}$, and a separately reported stress gauge. The stress gauge assigns $\sqrt{10}$ to the bottom half of calibration Taylor-to-activation ratios and its reciprocal to the rest; it uses no held-out effects. Primary judgments apply all three layer transformations jointly and recompute calibration scores and interventions in that common representation. The joint protocol was fixed during initial layer controls, before rank comparisons were inspected. Fixed transported masks separate reselection from changing an intervention; full-vocabulary probes use 32 test prompts, while margins use all 1,024. Masks and splits are retained with all seeds in the supplementary companion.

Let $E_\ell = \mathbb{E}[m^{\mathrm{dense}} - m^{\mathrm{ablated},\ell}]$. A supported ordering reversal requires opposite signs of $E_a - E_b$ with both simultaneous 95% intervals excluding zero. Paired, template-stratified bootstrap resampling (5,000 draws, seed 263001) jointly covers all 432 declared contrasts: 54 base and 378 non-base, across three checkpoints, three selectors, two budgets, and three layer pairs. Uncertainty is conditional on these checkpoints and fixed name/template pools. Rendered calibration/test prompts contain 127/507 unique counterbalanced pairs; the frozen weighting includes repetitions. Equal weighting of unique test pairs, with the same contrast family and seed 263002, preserves every supported-reversal flag.

Stored-weight and transport controls. Independent FP64 checks verify the ideal dyadic up/down maps exactly for all nine stored MLP layers, with gates unchanged. The module identities certify the real-arithmetic function of the stored pair, including joint composition; dense and fixed-mask dyadic margins are separately bit-identical on recorded probes. Joint and isolated masks agree in 504/504 comparisons; the maximum fixed-mask margin residual across the full grid is 1.4×10^{-4}. Probe agreement alone is not an all-input certificate. Reused evaluations for identical transported and reselected masks are explicitly recorded.

F.2 Held-out outcomes

For Qwen 1.5B, activation-based removal of the top 5% of channels gives early/middle (layers 4/14) mean margin losses $0.574890/0.179048$ at base and $0.290725/0.550656$ after the dyadic gauge. The early-minus-middle contrast reverses from $+0.395843$ to -0.259931, with simultaneous 95% bands $[0.232382, 0.559304]$ and $[-0.412536, -0.107326]$. The inference is about this selected-group margin-loss ordering, not a unique circuit or every meaning of layer importance. Candidate-name accuracy order does not reverse.

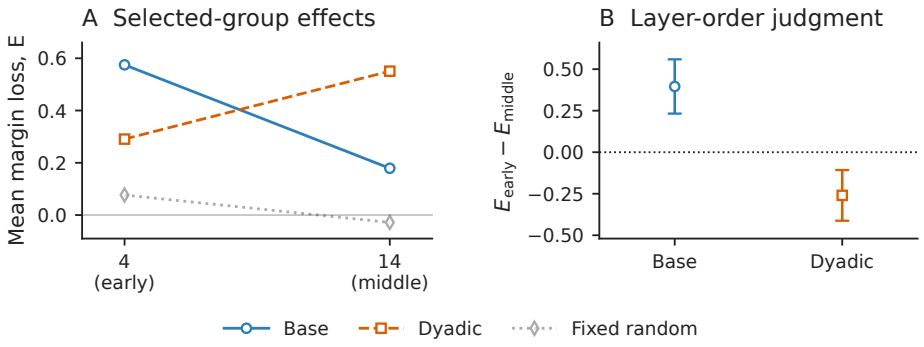


Figure 6: The selected-group margin-loss ordering reverses under activation-based ablation, while composed and Taylor controls preserve their selections and orderings. The example is included in the full simultaneous comparison family (Table 12); candidate-name accuracy ordering is unchanged.

Candidate-name accuracy after early/middle ablation is 98.73/99.80% at base and 99.22/99.71% after the dyadic gauge. Fixed random-mask mean losses are $0.076719/-0.028227$ in both frames; these are single matched masks per layer. Across the full grid, composed and Taylor selectors have transported Jaccard one and no reversals. All 11 supported activation reversals persist after deduplication. The full settings, outputs, controls, and sensitivity records are linked from the supplementary companion (Appendix D.12).

G Application Audits: Claims, Protocols, and Interpretation

The additional experiments ask what a researcher would conclude from an analysis of the same dense function. The three applications audit component-importance interpretations, model-edit acceptance, and pruning allocations. Their checkpoints, gauges, and procedures are specified separately from the original studies.

G.1 Which scientific claim is being compared?

Let $H(\theta)$ be an analysis output in the declared common frame, and let d report a decision from it. Proposition 2 applies separately to H and to $d \circ H$. Invariance of H implies invariance of $d \circ H$ for a fixed d, but the converse need not hold: different outputs can support the same decision. For labeled outputs, this argument requires that d respect the prescribed transport. It does not license choosing a favorable comparison after seeing the results.

This elementary consequence is operationally useful. A researcher reporting a most-important layer should test the layer choice as well as its underlying unit masks. An interpretation of pruning allocations as an importance map needs allocation comparisons in addition to final NLL. Edit acceptance based on rewrite, generalization, and locality needs a verdict comparison in addition to rewrite success. The applications distinguish these claims:

Application	Intermediate result	Reported decision and observed outcome
Component importance	Selected units and ablation margins	The top candidate layer stays unchanged despite activation-mask changes.
Pruning allocation	Reconstruction costs and sparse weights	Native SparseGPT* changes layer budgets, often compensating reconstruction differences in final NLL.
Model editing	Edited weights and target likelihoods	Native FT changes accept/reject decisions despite unchanged rewrite success.

A coordinate-dependent optimization geometry or compression representation can be a valid design choice. The audit identifies when a conclusion requires that convention to be stated, and tests re-

pairs under the same declared comparison. Improved consistency alone establishes neither causal faithfulness nor better pruning or editing utility.

G.2 Shared transformations and numerical controls

The confirmatory checkpoints are Qwen2.5-1.5B (28 layers, hidden width 1,536, MLP width 8,960) and TinyLlama-1.1B-Chat (22 layers, hidden width 2,048, MLP width 5,632; Zhang et al., 2024). Qwen2.5-0.5B is used for development diagnostics. Revision prefixes are `8faed761d45a`, `fe8a4ea1ffed`, and `060db6499f32`, respectively; full revisions and file hashes are in the model manifest.

For a SwiGLU block, write $h = \sigma(W_{\text{gate}}x + b_{\text{gate}}) \odot (W_{\text{up}}x + b_{\text{up}})$ and its output as Wh, where $W = W_{\text{down}}$. For a positive diagonal D,

$$(W_{\text{up}}, b_{\text{up}}, W) \mapsto (DW_{\text{up}}, Db_{\text{up}}, WD^{-1}) \quad \text{gives} \quad h' = Dh, \qquad W'h' = Wh.$$

The gate is unchanged. This algebraic identity certifies the real-arithmetic input–logit function on the full domain. Each checkpoint has a base and four fixed dyadic gauges D1–D4, with channel factors in $\{1/2, 1, 2\}$. Gauge seeds are 801101–801104, offset by 1000ℓ for layer ℓ. The tested tensors satisfy the ideal dyadic parameter map exactly when checked in FP64. Calculations use FP32 with TF32 disabled; dense probes check implementation rather than proving the all-domain identity.

For balanced methods, the channel scale is

$$b_i = \sqrt{\frac{\|W_{:i}\|_2}{\|[W_{\text{up},i:}, b_{\text{up},i}]\|_2}}.$$

Norms are computed in FP64 and the scale is cast to FP32. Zero-norm channels are rejected; none occurs here. The procedure acts in the balanced frame and its output is transported back. In exact arithmetic, $b'_i = b_i/d_i$ cancels the gauge. This normalizes the positive-diagonal orbit, not the full functional fiber.

Dyadic dense comparisons have zero observed logit gap on recorded probes. Balancing can introduce rounding relative to the original dense FP32 computation: maximum probe gaps are 2.86×10^{-5}/1.84×10^{-5} for editing execution controls and 3.79×10^{-5}/3.53×10^{-5} for pruning (Qwen/TinyLlama). Additional editing-case probes reach 4.39×10^{-5}/2.67×10^{-5}. These residuals must be distinguished from comparisons between gauges within a balanced procedure.

The critical procedure control transports a *fixed* edited or sparse output instead of independently recomputing it; its recorded output gap is zero. Independently recomputed Wanda and balanced SparseGPT* outputs also have zero common-frame weight difference and mask Jaccard 1 in the allocation experiment, whereas native SparseGPT* does not (Appendix H.4). This distinguishes transport from recomputation of a different output.

G.3 Development, confirmation, and uncertainty

Development selects eligibility, editor layers, and settings before locally frozen confirmation protocols. These are timestamped local freezes, not external preregistration. The CounterFact old-target preference screen is fixed before confirmation. Development and confirmation items are disjoint under the stated split rules. Reuse of BLiMP across importance and allocation experiments and WikiText across editing and allocation experiments, as well as reuse of checkpoints and gauges, precludes treating the applications as independent trained-model replications.

We use 5,000 paired bootstrap draws with seed 801900. Units are minimal pairs (stratified by BLiMP subtask), editing requests, or WikiText articles. Gauge/calibration conditions are jointly indexed contrasts, not independent trained models. Simultaneous intervals use the maximum absolute standardized deviation with the original standard errors. These are approximate bootstrap intervals, not exact finite-sample certificates. For claims selected from any of the three applications, we additionally use a per-application error budget of $0.05/3$.

Component importance. The primary family contains 240 ablation-margin contrasts. Resampling is paired within six subtasks. An order reversal requires opposite signs for both base and transformed layer contrasts, with both simultaneous intervals excluding zero. The practical threshold is 0.1 times the checkpoint's development standard deviation of the dense margin. A secondary family of 64 selected-layer accuracy contrasts is reported with a joint 304-contrast correction, distinguished from the original primary family.

Model editing. For request i, the rate averages four gauge/base disagreements before averaging requests. The 62-estimate family comprises 24 disagreement rates, eight native-minus-balanced consistency contrasts, and 30 balanced-minus-native utility contrasts. The case union has 319 requests, with 81 shared between checkpoints; shared case weights preserve this dependence. Each checkpoint contributes 200 cases. Zero observed disagreement is not proof of a zero population rate.

The inference implementation was conservatively corrected before full confirmatory results and family estimates were inspected, while some jobs had completed. Half of each family error budget α goes to nondegenerate maximum-statistic bootstrap intervals; half goes to zero-event bounds with a Bonferroni divisor of 62. For n independent request clusters with zero events, the one-sided upper bound is

$$u_n = 1 - \left(\frac{\alpha}{2 \cdot 62}\right)^{1/n}.$$

Bounding the probability of any within-request event also bounds its mean four-gauge rate. All-zero paired differences receive $[-u_n, u_n]$. For $n = 200$, the rate bound is 3.833% at $\alpha = 0.05$ and 4.359% at $\alpha = 0.05/3$. No cases, hypotheses, endpoints, or editor settings changed; original and corrected intervals are retained. The four gauges are not counted as 800 independent requests.

Pruning: language modeling. The family contains 276 contrasts: 96 gauge/base, 120 method-pair, and 60 adaptive/uniform differences. NLL is total loss divided by scored tokens; articles are resampled with shared weights across conditions. Model tokenizations use 33 and 31 test articles, with 130,560 scored tokens each; the article union has 33 clusters. Bands are conditional on the checkpoints, calibration samples, and gauges. A method-order reversal requires both method differences to exclude zero with opposite signs; a significant difference-of-differences alone is insufficient. Maximum-statistic critical values are 3.5649 within the allocation family and 3.9833 after the three-application correction.

G.4 Component-importance interpretation on BLiMP

BLiMP evaluates linguistic phenomena through minimally different acceptable/unacceptable sentence pairs (Warstadt et al., 2020). We use six subject–verb agreement subtasks: the two distractor-agreement subtasks (relational noun and relative clause) and versions 1/2 of irregular-plural and regular-plural subject–verb agreement. One duplicate unordered normalized pair is removed globally. Each subtask contributes 128 development, 128 calibration, and 512 test pairs, totaling 768/768/3,072 (seed 801201). All six are retained, including the harder relative-clause subtask. Test items remain within these templates and vocabulary; this is not unseen-template generalization.

The margin is acceptable-minus-unacceptable full-sequence log likelihood. Scores use calibration data, excluding padding, without sequence truncation. Candidate layers are $[4, 14, 23]$ for Qwen and $[3, 11, 18]$ for TinyLlama (zero-based). We select and zero the largest-scoring 5% or 10% of MLP channels with stable index tie-breaking. Scores are activation magnitude $a_i = \mathbb{E}|h_i|$, composed magnitude $a_i \|W_{:i}\|_2$, weighted input-feature variance (WIFV) $\mathrm{Var}(h_i)\|W_{:i}\|_2^2$, and signed Taylor influence. WIFV is a score used in FLAP (An et al., 2024); we do not run the complete FLAP pipeline. Taylor averages $h_i \, \partial m/\partial h_i$ over tokens and calibration pairs, where m is the pair margin; no absolute value is applied. The top candidate layer maximizes calibration mean dense-minus-ablated margin, with lexicographic tie-breaking. Ten random masks per layer/budget, fixed under transport, provide controls.

Activation mask Jaccard reaches 0.2291/0.2293 (Qwen/TinyLlama); composed, Taylor, and WIFV masks retain Jaccard 1. Two Qwen activation-induced order reversals survive the three-application correction, both between layers 14 and 23: base -0.0819 to $+0.1058$ at 5%/D3, and -0.1143 to $+0.0852$ at 10%/D4. Neither reaches the practical threshold (0.3792 Qwen; 0.2778 TinyLlama).

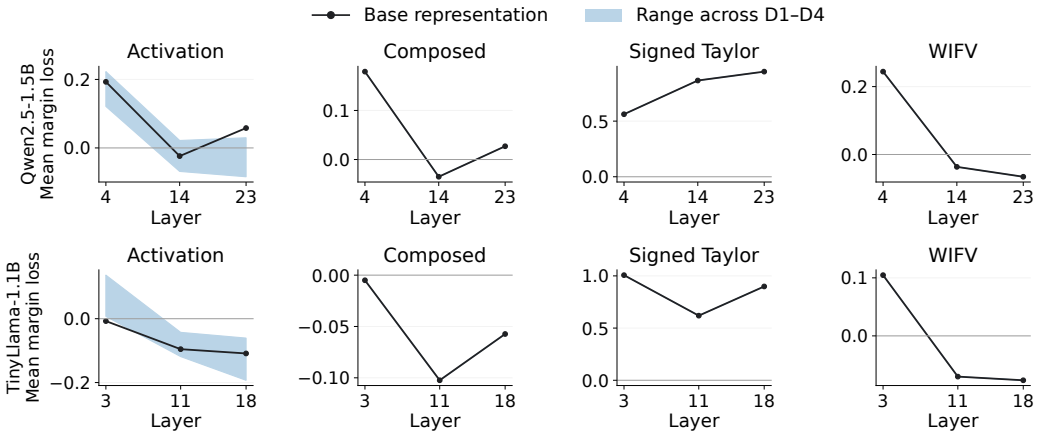


Figure 7: Component importance: 5% ablations across the three candidate layers. Activation masks alter importance margins while composed, Taylor, and WIFV controls remain consistent; the top calibration-selected candidate layer stays fixed. Shading shows the minimum–maximum range across four transformed parameterizations (D1–D4), excluding the base parameterization shown by the black line. Each value averages the same 3,072 held-out sentence pairs. Shading is not a confidence interval. The 10% results are summarized above and their full plot is in the supplementary companion.

TinyLlama has no supported reversal. The calibration-selected top candidate layer is unchanged for every checkpoint, score, budget, and gauge.

The 64 secondary accuracy contrasts have no nonzero interval after the joint correction; the largest absolute point change is 0.879 percentage points. Absence of significance is not an equivalence proof. This bounded result prevents mask instability from being promoted automatically to a changed top-layer conclusion. It establishes neither global layer rankings nor causal faithfulness of invariant scores.

H Pruning Allocations and the Quality of the Resulting Model

H.1 Allocation and evaluation protocol

We audit all 28/22 MLP down projections in Qwen/TinyLlama, without pruning attention. Each layer may receive sparsity 0, 25, 50, or 75%, subject to exactly 50% mask sparsity across the audited weights. Layer dimensions are constant within each checkpoint, so an exact quarter-budget dynamic program suffices. This masks 12.48%/11.53% of total model parameters, not 50% whole-model sparsity or a measured speedup.

Each calibration seed (801401/801402) supplies 262,144 WikiText-2 training tokens. For each method, representation, and layer, we profile three nonzero sparsities by pruning that layer in isolation and measuring NLL increase on 16,384 validation tokens (seed 801410, odd article-hash half). Zero sparsity has zero cost. The adaptive allocator minimizes the sum of these isolated-layer costs with deterministic tie-breaking; uniform allocation sets each audited layer to 50%. Both policies then construct a full sparse model sequentially, re-estimating the moment at each layer on the actual sparse prefix. The model is restored before each policy. Test evaluation uses a separate 131,072-token WikiText-2 stream (seed 801411), in 256-token windows, with 130,560 scored next-token positions. BLiMP accuracy on the 3,072 held-out component-importance pairs is secondary and is not used for allocation.

The allocator and validation criterion are our common wrapper. We do not attribute this algorithm to SparseGPT or Wanda, nor claim improved deployment performance. Full-model testing matters because a sum of isolated-layer costs need not predict interactions among pruned layers.

Table 13: Pruning allocation changes across eight gauge/calibration comparisons per checkpoint. "Layers" is the range of changed layer counts. Mean and maximum describe mean absolute sparsity changes across all audited layers, in percentage points (pp). The last column applies the prespecified 5 pp threshold.

Checkpoint	Method	Changed	Layers	Mean (pp)	Max (pp)	≥ 5 pp
Qwen 1.5B	SparseGPT*	8/8	2–4	2.902	3.571	0/8
Qwen 1.5B	Wanda	0/8	0–0	0.000	0.000	0/8
Qwen 1.5B	Balanced SparseGPT*	0/8	0–0	0.000	0.000	0/8
TinyLlama 1.1B	SparseGPT*	8/8	2–8	5.682	9.091	4/8
TinyLlama 1.1B	Wanda	0/8	0–0	0.000	0.000	0/8
TinyLlama 1.1B	Balanced SparseGPT*	0/8	0–0	0.000	0.000	0/8

Pruning implementations. SparseGPT* denotes our exact-cardinality adaptation of SparseGPT (Frantar & Alistarh, 2023). It retains sequential reconstruction and scalar damping (0.01), using stable rank selection within 128-column blocks to meet the mask quota exactly. The pinned upstream inclusive cutoff, `score <= sorted[floor(n*s)]`, removes at least one extra weight per block relative to that quota, potentially more under ties. Our moment is the same sum-of-outer-products estimator, scaled by $2/N_{\text{sequences}}$, with different FP32 accumulation order from the upstream online loop. These runs are not a bitwise reproduction of the complete upstream pipeline. A Qwen-0.5B development sensitivity check uses six layers and the unmodified upstream `fasterprune` kernel with our moment estimate; allocation changes also occur. This does not reproduce the official whole-model wrapper.

Wanda uses $|W_{ij}|\sqrt{H_{jj}}$ with an exact per-row quota (Sun et al., 2024). Balanced SparseGPT* normalizes each layer before profiling and assembly and transports final weights back. Reciprocal factors cancel in Wanda's score. SparseGPT's scalar damping does not transform covariantly under a general positive diagonal gauge, as derived in Appendix E. The new experiment traces this dependence into decisions about layer budgets.

H.2 Allocation changes and compensation

Native SparseGPT* changes allocations in all eight comparisons per checkpoint (Table 13). Qwen changes two to four layers and remains below the practical threshold. TinyLlama changes two to eight layers, exceeding the threshold in four comparisons. Wanda and balanced SparseGPT* preserve every allocation. Figure 8 shows the full grid.

At uniform allocation, all native SparseGPT* gauge/base NLL point differences are positive, with maxima 0.013691/0.004407 (Qwen/TinyLlama). After both the 276-contrast and three-application corrections, seven/eight Qwen and five/eight TinyLlama differences exclude zero. At adaptive allocation, zero/eight exclude zero for either model. This does not prove adaptive quality equivalence.

Let $L(g, a)$ denote test NLL after reconstruction in representation g with allocation a, and a_b, a_g the selected base/gauge allocations. Additional fixed-base-allocation models permit the diagnostic

$$L(g, a_g) - L(b, a_b) = \underbrace{L(g, a_g) - L(g, a_b)}_{\text{allocation change at fixed gauge}} + \underbrace{L(g, a_b) - L(b, a_b)}_{\text{reconstruction at fixed allocation}} . \tag{23}$$

This is an exact algebraic, reference-dependent diagnostic, not a unique causal decomposition. In 15/16 native SparseGPT* comparisons the allocation term is negative; in all 16 the reconstruction term is positive (Table 14). For Qwen, calibration seed 801401 and the first fixed rescaling (D1) give base adaptive NLL 2.738009, gauge with fixed base allocation 2.744571, and gauge with newly selected allocation 2.737692. Figure 2 presents this first calibration/gauge condition for both checkpoints. The allocation change offsets reconstruction differences. It cannot be read simply as harmful instability, and close final NLL cannot identify a stable layer-importance map.

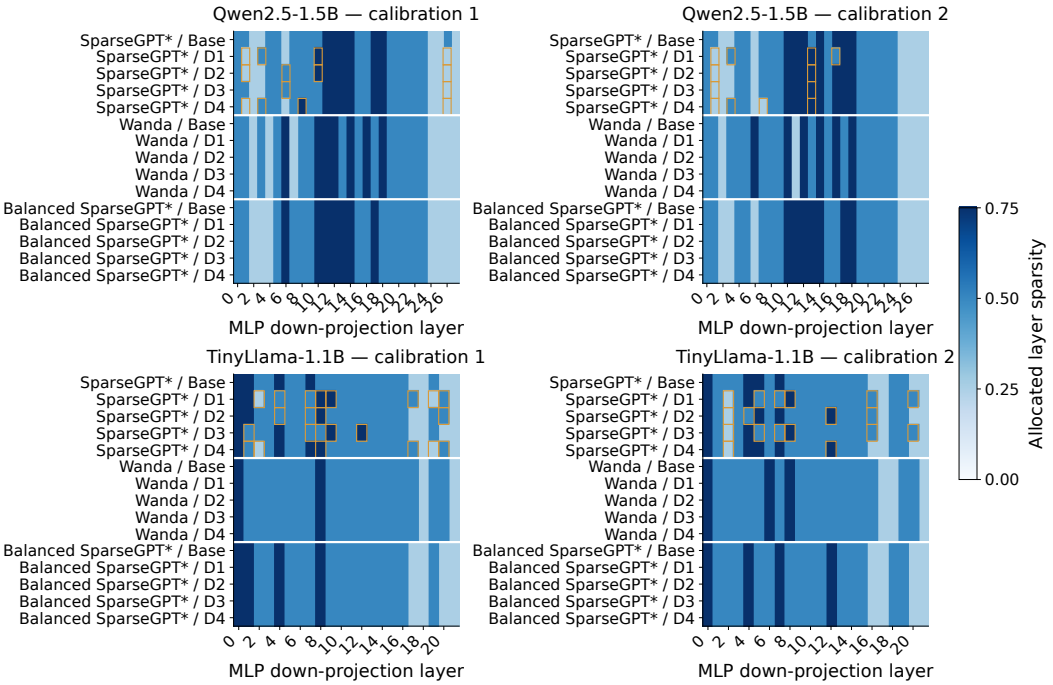


Figure 8: Complete layer allocations. The labels D1–D4 denote four fixed dyadic rescalings. Outlines mark sparsities that differ from the same method/calibration's base allocation. Every row has the same 50% budget within MLP down weights. Choices can differ even when final quality is close.

Table 14: All 16 native SparseGPT* allocation/reconstruction diagnostics. The last two columns sum to the total gap in Equation 23, before rounding. Calibration 1/2 denotes seed 801401/801402.

Checkpoint	Cal.	Gauge	Total NLL gap	Allocation term	Reconstruction term
Qwen 1.5B	1	D1	-0.000317	-0.006879	+0.006562
Qwen 1.5B	1	D2	-0.002507	-0.008592	+0.006084
Qwen 1.5B	1	D3	-0.001337	-0.004059	+0.002722
Qwen 1.5B	1	D4	+0.000646	-0.005463	+0.006110
Qwen 1.5B	2	D1	+0.001162	-0.003901	+0.005063
Qwen 1.5B	2	D2	+0.002875	-0.002730	+0.005605
Qwen 1.5B	2	D3	+0.000240	-0.000976	+0.001215
Qwen 1.5B	2	D4	+0.002746	-0.002516	+0.005261
TinyLlama 1.1B	1	D1	-0.000132	-0.009437	+0.009304
TinyLlama 1.1B	1	D2	-0.000351	-0.004007	+0.003656
TinyLlama 1.1B	1	D3	-0.002823	-0.012663	+0.009840
TinyLlama 1.1B	1	D4	-0.003193	-0.006063	+0.002871
TinyLlama 1.1B	2	D1	-0.002575	-0.011093	+0.008518
TinyLlama 1.1B	2	D2	-0.000110	-0.002533	+0.002423
TinyLlama 1.1B	2	D3	-0.002400	-0.009965	+0.007565
TinyLlama 1.1B	2	D4	+0.001365	+0.000798	+0.000568

H.3 Quality comparisons and limits

Six point-estimate method-order reversals occur, all for native versus balanced SparseGPT* in TinyLlama/calibration 801402. One survives simultaneous inference and the three-application correction: under uniform allocation, native minus balanced NLL is -0.001797 at the base, with interval $[-0.002936, -0.000658]$, and $+0.002610$ at D1, with interval $[+0.0000183, +0.005201]$.

Table 15: Held-out NLL for all model/method/calibration groups. Each gauge range summarizes D1–D4, not uncertainty; inference uses paired article bootstrap. Calibration 1/2 denotes seed 801401/801402; SG* abbreviates SparseGPT*. All 120 individual quality records, including secondary BLiMP accuracy, are retained.

			Uniform NLL		Adaptive NLL	
Checkpoint	Cal.	Method	Base	D1–D4 range	Base	D1–D4 range
Qwen	1	SG*	2.753698	2.758394–2.767389	2.738009	2.735502–2.738656
Qwen	1	Wanda	2.792034	2.792034–2.792034	2.783724	2.783724–2.783724
Qwen	1	Balanced SG*	2.750503	2.750503–2.750503	2.733927	2.733927–2.733927
Qwen	2	SG*	2.753734	2.758889–2.766017	2.733949	2.734189–2.736824
Qwen	2	Wanda	2.791992	2.791992–2.791992	2.781401	2.781401–2.781401
Qwen	2	Balanced SG*	2.751234	2.751234–2.751234	2.732099	2.732099–2.732099
TinyLlama	1	SG*	2.664736	2.667000–2.668000	2.672147	2.668954–2.672014
TinyLlama	1	Wanda	2.684795	2.684795–2.684795	2.682705	2.682705–2.682705
TinyLlama	1	Balanced SG*	2.664249	2.664249–2.664249	2.672322	2.672322–2.672322
TinyLlama	2	SG*	2.663575	2.666414–2.667982	2.668527	2.665952–2.669893
TinyLlama	2	Wanda	2.685618	2.685618–2.685618	2.685917	2.685917–2.685917
TinyLlama	2	Balanced SG*	2.665372	2.665372–2.665372	2.667099	2.667099–2.667099

Native NLLs are 2.663575 and 2.667982; balanced NLL is 2.665372 at both. This is a small effect with one endpoint near zero. We observe no SparseGPT*/Wanda order reversal and do not claim broad ranking failure of established pruning methods. The supplementary companion reports all 40 native/balanced contrasts and their corrected intervals.

Adaptive allocation has higher test NLL than uniform in 23/60 point comparisons. This includes repeated identical gauge outcomes for consistent methods, not 60 independent model trials. It prevents interpreting this as a superior allocator. A post-result exploratory summary selects a method by validation NLL within each fixed allocation policy: the choice changes in 12/32 non-base comparisons (Qwen 3, TinyLlama 9). This is not a new confirmatory test or an independent sample.

H.4 Sparse-output controls

Transporting a fixed sparse output gives zero observed logit gap for all methods; two reloaded test windows reproduce NLL exactly. Independently constructed Wanda and balanced SparseGPT* outputs have zero common-frame weight difference, mask Jaccard 1, and identical evaluated NLL and BLiMP accuracy. Native SparseGPT* instead has minimum mask Jaccard 0.8732/0.8067, maximum relative weight L_2 gap 0.2313/0.2729, and maximum full-logit probe gap 12.3383/6.0637 (Qwen/TinyLlama). These last maxima describe 64-token probes, not all inputs. Streaming versus full NLL differs by zero for Qwen and at most 2.38×10^{-7} for TinyLlama. Probe, weight, mask, and held-out quality checks support different statements.

An allocation should therefore be qualified as a decision of a specified procedure and coordinate convention unless its claimed representation independence has been checked. The audit also distinguishes reconstruction changes at a fixed allocation from choosing another allocation. It does not require abandoning a useful coordinate-dependent compressor.

I Model-Edit Acceptance under Exact Reparameterization

I.1 Cohort and declared decision rule

CounterFact accompanies ROME and provides rewrite requests, paraphrases, and neighborhood prompts (Meng et al., 2022). We select one request per normalized subject string (minimum case ID), hash-order subjects, and separate development and confirmation pools. The development pool contains 1,024 subjects, yielding 64 eligible cases per checkpoint; the confirmation pool contains 19,367 subjects. The first 200 eligible confirmation cases per checkpoint form cohorts with 81 shared cases. Subject strings are disjoint across development/confirmation; external alias-resolved entity separation is not claimed.

Eligibility requires single-token old and new targets in the original prompt and a higher base likelihood for the old target. The screen was fixed before confirmation. It does not require the old target to be global top-1: only 43/200 Qwen and 45/200 TinyLlama cases satisfy that stronger condition. The result therefore concerns requests with an old-over-new preference, not exclusively facts previously recalled correctly at top-1. An exploratory restriction to the stronger subgroup finds at least one acceptance change in 2/43 and 3/45 cases; it is not a separately powered confirmatory result.

Each case has two paraphrases and ten neighborhood prompts. Alternative prompts use teacher-forced full-target log likelihood when targets span several tokens, with prefix/tokenization checks. Rewrite and paraphrase success mean new-over-old preference. Among neighborhoods originally preferring old over new, a flip reverses that preference after editing. The primary rule is

$$\text{accept} = (\text{rewrite succeeds}) \wedge (\text{paraphrase success} \geq 0.8) \wedge (\text{eligible neighbor flips} = 0).$$

With two paraphrases, 80% requires both. Cases without eligible neighborhoods are unassessed, not silently accepted. All 6,000 confirmatory edits have zero failures and zero unassessed outcomes. Global top-1 rewrite and paraphrase success are separate utility diagnostics, not substitutes for the preference rule. The top-1 diagnostic counts a success only when the target is one token and equals the global next-token argmax; multi-token targets do not count as top-1 successes. This is an operational benchmark criterion, not a universal deployment policy.

I.2 Editor implementations and controls

Every edit starts from fresh checkpoint weights. Development tests 64 cases at three candidate layers with two native editors per checkpoint (768 edits). A common layer maximizing average native FT/ROME development quality is layer 4 for Qwen and layer 3 for TinyLlama (zero-based). Confirmatory outcomes select neither layers nor hyperparameters.

Constrained fine-tuning. Native FT optimizes one MLP down-projection matrix with Adam, learning rate 5×10^{-4}, at most 25 steps, no weight decay, scalar entrywise L_∞ displacement bound $\epsilon = 5 \times 10^{-4}$, and stopping at rewrite cross-entropy below 0.01. Scalar settings are fixed across representations. Balanced FT applies the shared channel normalization, edits there, and transports the output back. It is a distinct procedure with separately measured utility.

With $h' = Dh$ and $W' = WD^{-1}$, corresponding edits require $\Delta W' = \Delta W D^{-1}$. The base box $|\Delta W_{ij}| \leq \epsilon$ corresponds to $|\Delta W'_{ij}| \leq \epsilon/d_j$. The same scalar bound in both frames changes the functional set of feasible edits. Preserving only the box also fails to transport Adam's coordinate-wise trajectory. A base-frame optimization variable $U = W'D$, together with the transported box, restores correspondence in a development control; transporting only one does not. A 32-case pilot has zero rewrite/acceptance flips despite likelihood changes. This no-effect pilot is retained and was not used to tune a desired flip rate.

Native FT dependence is thus interpretable from its specification. The audit tests whether it reaches a claimed acceptance decision and whether a specified repair changes it. Coordinate-dependent optimization is not intrinsically invalid; a function-level interpretation needs its geometry or transport declared.

ROME adaptation. We use a single-layer rank-one update with a subject-final-token key, 21 fixed development-generated contexts, and an optimized MLP-output displacement. Settings are 20 steps, learning rate 0.5, KL weight 0.0625, norm penalty 0.5, and clamp factor 4. Each representation's fresh uncentered moment uses 262,144 WikiText-2 training input tokens (seed 801302, context length 256), FP32 accumulation, and FP64 unregularized Cholesky. No ridge or stale moment cache is used. This adapts the ROME implementation to the declared checkpoints; it does not stand for every ROME implementation.

For a nonsingular moment C and key k, let $u = C^{-1}k$. Under $C' = DCD$ and $k' = Dk$, $u' = D^{-1}u$. If output-space target displacement r and its optimization are unchanged,

$$\Delta W = \frac{ru^\top}{u^\top k_*}, \qquad \Delta W' = \frac{r(u')^\top}{(u')^\top D k_*} = \Delta W D^{-1},$$

where k_* is the implementation's normalization key. This depends on matching keys, output objectives, fresh covariant moments, and a nonsingular unregularized solve. An untransported ridge or

Table 16: Model-edit mean paired disagreement (%) and simultaneous 95% intervals for the 62-estimate family. Each of 200 requests averages four gauges; paraphrase and neighborhood rates also average within-request prompts. Balanced FT and ROME each give $0\,[0, 3.833]$ for every endpoint. These are consistency measures; utility is reported separately.

Checkpoint	Endpoint	Native FT [simultaneous 95% interval]
Qwen 1.5B	Rewrite	0.000 [0.000, 3.833]
	Paraphrase	7.312 [3.329, 11.296]
	Neighborhood	2.152 [0.731, 3.574]
	Acceptance	8.125 [2.337, 13.913]
TinyLlama 1.1B	Rewrite	0.000 [0.000, 3.833]
	Paraphrase	4.625 [0.971, 8.279]
	Neighborhood	1.011 [0.201, 1.821]
	Acceptance	7.875 [1.869, 13.881]

Table 17: Base-representation utility, separate from consistency. Rates are percentages; the last column counts accepted edits. Paraphrase preference compares new/old likelihood; top-1 checks the global next-token choice. Locality retention uses originally old-preferred neighbors.

		Rewrite	Paraphrase		Locality	Accept
Checkpoint	Method	top-1	Preference	top-1	retained	/200
Qwen 1.5B	Native FT	100.0	65.25	25.50	90.19	65
Qwen 1.5B	Balanced FT	100.0	65.25	25.75	90.17	66
Qwen 1.5B	ROME adaptation	100.0	97.50	76.25	95.31	159
TinyLlama 1.1B	Native FT	100.0	80.25	38.75	94.72	108
TinyLlama 1.1B	Balanced FT	100.0	80.25	38.75	94.72	108
TinyLlama 1.1B	ROME adaptation	80.0	89.00	52.75	98.76	156

incompatible cache can break it. It explains this control under the tested transformation; it is not a general new theorem about all editors.

I.3 Acceptance disagreement and task utility

Native FT achieves new-target top-1 throughout, but paired acceptance disagrees by 8.125/7.875% (Qwen/TinyLlama), affecting 21/200 and 18/200 unique requests. Table 1 gives within-family intervals; three-application intervals are [1.727,14.523]% and [1.237,14.513]%. Point rates exceed the prespecified 5 pp threshold, but lower confidence limits do not. Rewrite success therefore misses changes in the full verdict. Balanced FT and ROME have zero observed disagreement on all endpoints, with upper bounds 3.833% within the editing family and 4.359% across applications; ROME is a mechanistic positive control.

Table 17 reports utility separately. TinyLlama ROME reaches 92% rewrite preference but 80% global top-1, compared with 100% top-1 for both FT variants. Balanced FT's prespecified 2 pp noninferiority criterion holds in only 9/30 simultaneous utility contrasts; some gauges favor native paraphrase success by 3.0 pp in Qwen or 2.75 pp in TinyLlama. Balancing therefore improves consistency without establishing all-condition utility noninferiority.

I.4 Transport, replay, and threshold checks

All 6,000 updates have hashes; Table 18 gives the 4,800 common-frame comparisons. Balanced FT updates are identical, while ROME's tiny residuals leave evaluated likelihoods and decisions unchanged. Eight fresh moment checks satisfy $C_g = DCD$ exactly on stored inputs. Fixed-output transport distinguishes coordinate changes from independent recomputation.

Two post-result cases, selected in fixed gauge/cohort order, were rerun from fresh checkpoints with balanced controls (eight edits). Qwen CounterFact case 14083 loses one of two successful para-

Table 18: All 4,800 common-frame edit-update comparisons. "Exact" means identical compared update tensors. Small ROME residuals do not change evaluated likelihoods or decisions. Transport of a fixed edited output has zero observed gap for all methods.

Checkpoint	Method	Exact comparisons	Median relative L_2	Max relative L_2
Qwen 1.5B	Native FT	0/800	0.8124	1.57×10^{0}
Qwen 1.5B	Balanced FT	800/800	0.0000	0
Qwen 1.5B	ROME adaptation	786/800	0.0000	1.63×10^{-11}
TinyLlama 1.1B	Native FT	0/800	0.7140	1.25×10^{0}
TinyLlama 1.1B	Balanced FT	800/800	0.0000	0
TinyLlama 1.1B	ROME adaptation	799/800	0.0000	7.01×10^{-12}

phrases with stable neighbors; TinyLlama CounterFact case 14120 retains both but flips one of ten eligible neighbors. Native verdicts change from accept to reject; balanced FT accepts both frames. Replayed likelihoods match exactly. These are mechanism witnesses, not independent evidence or validation of replacement facts.

Table 19: Exploratory threshold sensitivity for native FT. The identical 80% and 100% rules are combined because each request has two paraphrases. Balanced FT and ROME have zero disagreement at all nine original threshold pairs. This sensitivity supplies no new confirmatory test.

Paraphrase minimum (%)	Neighbor tolerance (%)	Qwen disagreement (%)	TinyLlama (%)
50	0	10.000	6.750
50	10	8.500	4.500
50	20	6.625	4.500
80 or 100	0	8.125	7.875
80 or 100	10	7.875	5.125
80 or 100	20	8.375	6.000

Across the nine threshold pairs, native disagreement ranges from 6.625–10.000% in Qwen and 4.500–7.875% in TinyLlama; controls remain at zero. This exploratory sensitivity retains the frozen rule. Broader domain and generation-level evaluation is needed beyond these checkpoints and screened cohorts.