AS A HIDDEN HYPERPARAMETER, HARDWARE CHANGES THE TRAINING ALGORITHM

Anonymous authors
Paper under double-blind review

ABSTRACT

Splitting a training update across GPUs or microbatches should save time and memory, not change the algorithm. Yet in practice, it can. We introduce Partition Invariance: repartitioning the same data must preserve the training update in exact arithmetic when the specified algorithm is unchanged. We prove a sufficient condition for this property and use its premises to guide diagnosis. PartitionCheck tests this property by comparing the same update across two partition configurations, allowing for floating-point differences. It also checks each execution against an independent reference implementation, since matching executions can share the same error. Beyond detection, we characterize how partitioning changes the algorithm. Under stated conditions, constant gradient scaling changes the effective learning rate under SGD but weight decay under LARS. In five codebases, we test whether code-derived rewrites reproduce the affected runs. Across 31 frameworks and research codebases, including Transformers, DeepSpeed, and veRL, we identify 43 implementation errors; six affect every partition alike. These errors can undermine research conclusions even without degrading model quality. In SimCSE, two GPUs double the relative weight of an auxiliary loss. Fixed padding bypasses it, making an ablation compare identical training runs. In a preregistered experiment with a 134M-parameter Mamba-2 model, the affected tensor-parallel implementation yields 0.189 nats higher test loss than its repaired counterpart when both are evaluated under the declared architecture. Reporting the partition is not bookkeeping; it is part of reporting the method.

1 INTRODUCTION

Training algorithms are described through their objectives, architectures, and optimization rules. Implementing them also requires dividing each update across GPUs and microbatches, including padding and data placement. We call this division the partition of an update. These choices should change resource use, not the specified algorithm. In the released Barlow Twins implementation (Zbontar et al., 2021), however, doubling the GPU count halves the effective learning rate for biases and normalization parameters. Keeping the configured learning rate fixed therefore does not keep the algorithm fixed. A resource choice becomes a hidden hyperparameter.

The mismatch can also invalidate an ablation. In SimCSE (Gao et al., 2021), researchers can enable an auxiliary masked-language-modeling (MLM) objective to measure its contribution. Under fixed padding, however, the released code omits this objective even when enabled. In our experiments, enabling or disabling MLM produces identical saved checkpoints and evaluation scores. The ablation therefore cannot measure MLM's contribution. Thus, unchanged algorithm settings can yield different updates, while an ablation switch can leave the algorithm unchanged.

The mismatch can also affect conclusions about memory requirements. GaLore's released configuration for training a 7B model on one GPU in per-layer mode (Zhao et al., 2024) performs 32 optimizer updates where one is specified. In our experiments, it fits on a 24 GB GPU. Restoring the single update causes the same model to run out of memory unless gradients are stored in host memory. The configuration fits because the update rule changes.

These violations are easy to miss: training can converge and evaluation scores can look normal. In two paired Mask2Former continuations (Cheng et al., 2022), original and repaired implementations assign different classes to about 3% of pixels, yet endpoint panoptic quality differs by less than 0.04

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

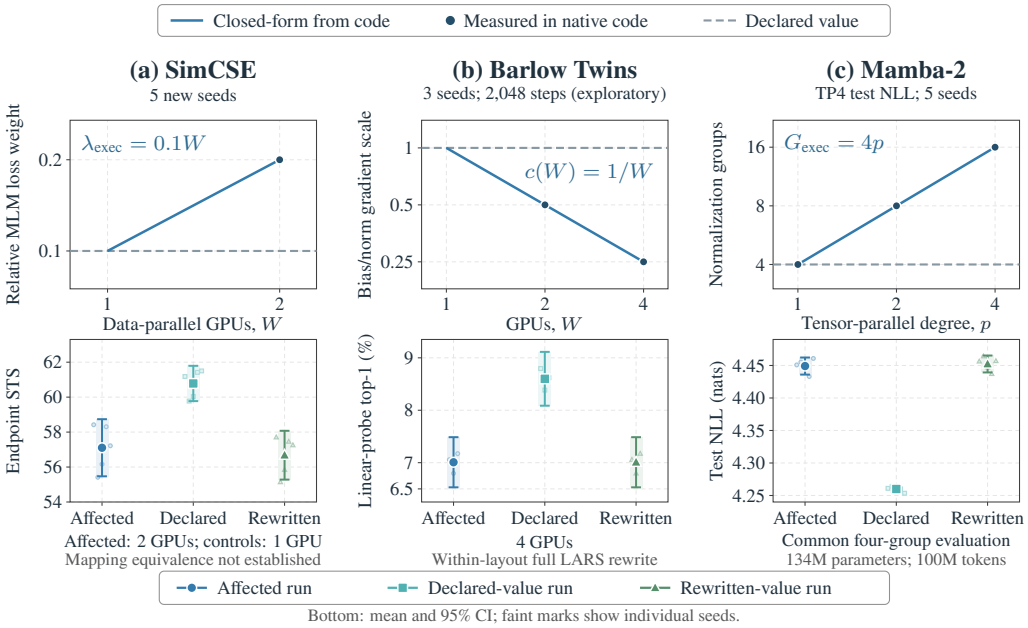

Figure 1: One declared algorithm, several executed algorithms. Each panel varies only a resource knob. Top: an algorithm variable's executed value, derived in closed form from code (line) and measured in native code (markers). Bottom: quality of affected runs and runs using declared or rewritten values. (a) SimCSE: weight of the MLM loss relative to the contrastive loss vs. data-parallel GPUs; endpoint STS, 5 new seeds. (b) Barlow Twins: gradient scale of bias and normalization parameters vs. GPUs; linear-probe top-1, 3 seeds, 2,048 steps (exploratory). (c) Mamba-2: normalization groups vs. tensor-parallel degree; TP4 test NLL under the declared architecture, 5 seeds.

points. Internal comparisons also require care: differences can reflect intentional choices or floating-point rounding, while agreement can hide a shared error. In Q-GaLore's released entry point (Zhang et al., 2024), seven floating-point parameters receive gradients but are never updated in any tested configuration. Comparing configurations therefore misses the error.

Existing approaches provide useful checks, but the comparison must first specify which executions should implement the same algorithm. Batch-invariant kernels address floating-point variation across batch sizes (He & Thinking Machines Lab, 2025); the changes studied here persist in exact arithmetic. Plan verifiers check parallelization strategies against a logical model (Lu et al., 2025), while training debuggers compare executions with references (Jiang et al., 2025a; Zhou et al., 2026) or inferred invariants (Jiang et al., 2025b). Our comparisons also cover code outside the model graph, including data collation, loss normalization, clipping, and optimizer and scheduler updates. Metamorphic testing (Chen et al., 2018) supplies the comparison principle. Applying it here requires defining admissible repartitions, accounting for numerical variation, and checking for shared errors.

We define a logical update by its examples, complete initial state, a contract specifying the intended computation, and a single state transition. Partition Invariance requires every admissible partition of this update to produce the same resulting training state in exact arithmetic. The contract distinguishes implementation errors from declared design choices. Our contributions are:

- **RQ1: How can we detect whether an implementation changes the intended algorithm?** (§2–§3, §5.1). Comparing the same global batch is not enough: the executions must represent the same logical update, including its complete initial state, intended update boundary, and computation outside the model graph. PartitionCheck compares two admissible partitions of one logical update under calibrated floating-point tolerances and checks each against an independent reference, separating agreement from correctness. It detects all 14 faults in the original evaluation and all 36 in a separate archival replay, with 0/9 and 0/36 benign alarms, respectively.

- **RQ2: How does the implementation change the algorithm, and by how much?** (§4.1–§4.2). We derive how errors change objectives, optimizer updates, and architectures, and test these derivations with explicit rewrites in five codebases. Under stated conditions, Proposition 2 maps constant gradient scaling to changes in the learning rate under SGD, weight decay for LARS-adapted parameters (You et al., 2017), or ϵ under Adam (Kingma & Ba, 2015). Scaling loss terms differently changes their relative weights. Other changes require code-specific analysis: Mamba-2's unfused tensor-parallel path (Dao & Gu, 2024) uses eight normalization groups on two GPUs when the declared architecture has four.
- **RQ3: How do these errors affect model training and research conclusions?** (§4.3, §5). We identify 43 errors across 31 codebases, including Transformers (Wolf et al., 2020), DeepSpeed (Rasley et al., 2020), and veRL (Sheng et al., 2025). Thirty-seven errors vary with the partition; six affect all tested partitions. They can invalidate ablations, alter memory requirements, or affect model quality (Figure 1). In a preregistered, five-seed experiment, a 134M-parameter Mamba-2 model trained with the affected four-way tensor-parallel implementation has 0.189 nats higher test loss than its repaired counterpart when both are evaluated under the declared architecture (97.5% CI $[0.170, 0.209]$). The gap is 0.053 nats when each is evaluated as trained.

Scope. Findings are specific to the audited versions, configurations, and contracts (cf. Appendix D); where semantics are unspecified, they depend on our stated contract. Finite tests can refute Partition Invariance, not establish it, and the audit identifies mechanisms, not prevalence. Section 7 covers limitations, prior public reports, and implications for published results.

2 Partition Invariance and What a Partition Can Rewrite

Comparing two runs is informative only after we specify which runs should agree. Definition 1 fixes the comparison; Proposition 1 gives sufficient conditions that guide diagnosis; and Proposition 2 maps constant gradient scaling to the algorithm executed. SimCSE connects these three steps.

2.1 Logical updates and invariance

We compare different ways of distributing the same training update. We define a *logical update* as $\mathcal{L} = (X_0, \mathcal{A}, \mathcal{C}, T)$. The initial state X_0 includes parameters, optimizer moments, learning-rate scheduler counters, the mixed-precision loss scale, model buffers, and random state. Parameters alone are insufficient: other state can change later updates even when the current parameters agree.

The inputs $\mathcal{A}$ are examples, sequences, or views with their identities, weights, masks, and token counts. The *contract* $\mathcal{C}$ specifies the objective, normalization, random choices, clipping, precision, and required global information. Contracts come from the paper, configuration, or documentation (cf. Appendix D). Where these leave a choice open, we state and test our interpretation. The transition T advances the complete training state, including the timing of auxiliary-state updates.

A *partition configuration* divides computation across microbatches and devices, including padding, accumulation, sharding, and reduction order. It is *admissible* if its plan preserves X_0, $\mathcal{A}$, $\mathcal{C}$, and T. Admissibility concerns the plan; execution is what we test. Declaring an optimizer step per microbatch describes a different algorithm from declaring one after accumulation; if the latter plan executes several steps, the plan remains admissible and the execution violates it. A plan is *non-comparable* if its declarations change the training rules, such as using per-GPU instead of full-batch negatives, and a *boundary* case if it omits global information required by a batch-coupled operation, such as the full gradient norm for global clipping. These checks precede execution (cf. Appendix A).

Let $U_p(\mathcal{L})$ denote the final training state and the intermediate quantities specified by $\mathcal{C}$ under partition p. Comparisons refer to the same mathematical quantities, such as the accumulated gradient, with sharded or reordered tensors aligned to a common logical representation.

Definition 1 (Partition Invariance)**.** An implementation satisfies Partition Invariance for a logical update $\mathcal{L}$ if $U_p(\mathcal{L}) = U_q(\mathcal{L})$ for every pair of admissible partition configurations p and q, in exact arithmetic retaining all specified operations, including quantization.

Agreement does not establish correctness: all partitions can share an error. PartitionCheck therefore separately checks *reference conformance*, agreement with the intended computation (§3).

2.2 WHERE A PARTITION LEAKS IN

Consider a token-mean loss. For sequence a, let N_a sum its token losses and m_a count supervised tokens. The objective is $(\sum_a N_a)/(\sum_a m_a)$. Each microbatch computes $S(\mathcal{A}_i) = (\sum_{a\in\mathcal{A}_i} N_a, \sum_{a\in\mathcal{A}_i} m_a)$, and the full-batch statistic is $S(\mathcal{A}) = \sum_i S(\mathcal{A}_i)$. Summing these pairs before division preserves the objective; averaging microbatch means need not.

Addition is associative (grouping does not matter) and commutative (order does not matter). With an identity element, these properties define a commutative monoid. Preserving the full update also requires consistent inputs, computation rules, and state updates.

Proposition 1 (Sufficient conditions for Partition Invariance)**.** Fix $\mathcal{L} = (X_0, \mathcal{A}, \mathcal{C}, T)$. Suppose each stage j reads the inputs only through a statistic S_j in a commutative monoid $(M_j, \oplus_j)$, with $S_j(\mathcal{A}) = S_j(\mathcal{A}_1)\oplus_j \cdots \oplus_j S_j(\mathcal{A}_m)$ for every admissible partition (*exact merge*). Partition Invariance holds if all of the following also hold:

(i) **Inputs.** Preserve every input field.
(ii) **Randomness.** Each draw is fixed by seed, update, input identity, operation, and view.
(iii) **Global information.** Each batch-coupled stage receives all information required by $\mathcal{C}$.
(iv) **Computation.** Each stage uses only X_0, merged statistics, and $\mathcal{C}$; ties are deterministic.
(v) **State update.** Apply the same deterministic T once per logical update, after all merges.

All admissible partitions then agree on the specified intermediate quantities and final training state.

The proof follows the stages in order: identical inputs and merged information give identical outputs (cf. Appendix B). The conditions are sufficient, not necessary. A violation must break a premise, but this alone does not locate the cause; the premises guide code inspection, and execution tests the predicted computation (§4). The audited violations fall into four mechanism classes (Table D.5). A *reduction scale* breaks exact merge: in SimCSE, each GPU backpropagates only through its local slice of the gathered contrastive batch before gradient averaging. A *local statistic* lacks global information: Mask2Former's rank-local classification denominators reweight images. A partition-dependent *operator or dispatch* changes the computation: Mamba-2 uses pG normalization groups at tensor-parallel degree p, and fixed padding selects a SimCSE collator that omits MLM, so two configurations that both omit it may agree yet fail reference conformance. A *clock* breaks the single state update: GaLore takes k optimizer steps per declared update of k accumulation steps.

2.3 WHAT A LEAK BECOMES

Once a violation is located, what algorithm runs? For constant gradient scaling, the answer depends on the optimizer: scaling a gradient by c need not scale the effective learning rate by c.

Proposition 2 (Optimizer-specific maps for a constant gradient scale)**.** Let two executions share initial parameters, inputs, randomness, and update boundaries, with data gradients $\widetilde{g}_t = c\, h_t$ at every common parameter state for a constant $c > 0$. In exact arithmetic from zero optimizer states, the $\widetilde{g}_t$ execution matches the h_t parameter sequence under the following settings (cf. Appendix B):

SGD with heavy-ball momentum	$\eta_t \mapsto c\,\eta_t$; coupled weight decay $\mu \mapsto \mu/c$
Adam, AdamW	$\epsilon \mapsto \epsilon/c$, learning rate unchanged; coupled weight decay $\mu \mapsto \mu/c$, decoupled weight decay unchanged
LARS as in Barlow Twins	adapted groups $\mu \mapsto \mu/c$; excluded bias and normalization groups, which have no weight decay, $\eta_t \mapsto c\,\eta_t$
Ideal global-norm clipping	$\tau \mapsto \tau/c$, composed with the maps above; an additive stabilizer is also divided by c

The LARS map requires that a nonzero adaptation vector is never evaluated at zero parameter norm. For unequally scaled loss terms, $\widetilde{g}_t = \sum_k a_k \nabla \ell_k = c \sum_k \lambda'_k \nabla \ell_k$ with $\lambda'_k = a_k/c$ defines a mapped objective to which the same maps apply.

These maps compare parameter trajectories under adjusted hyperparameters, not optimizer states, which can differ by powers of c (cf. Appendix B.3); Partition Invariance compares complete states under an unchanged contract. We use established optimizer scale relations (Kingma & Ba, 2015; You et al., 2017) to explain violations and make testable predictions.

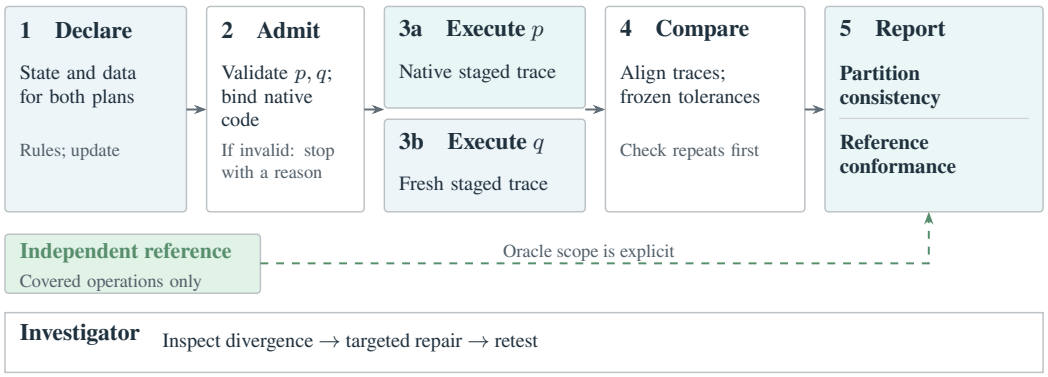

Figure 2: PartitionCheck workflow. For one logical update, the checker validates two plans and compares aligned native traces under frozen tolerances. It separately reports partition consistency and reference conformance for covered operations; investigators analyze and repair discrepancies.

For SimCSE, the intended objective is $\ell_{\text{con}} + \lambda \ell_{\text{mlm}}$. On W data-parallel GPUs, the local contrastive backward pass and gradient averaging give $\widetilde{g} = \frac{1}{W} g_{\text{con}} + \lambda g_{\text{mlm}} = \frac{1}{W}(g_{\text{con}} + W\lambda g_{\text{mlm}})$. Thus the relative MLM coefficient is $W\lambda$: two GPUs turn $\lambda = 0.1$ into 0.2. Under AdamW, the common scale $1/W$ also corresponds to using $W\epsilon$ with this mapped objective.

In Barlow Twins, the non-differentiable all-reduce followed by gradient averaging scales every loss term by $1/W$. Under the audited LARS implementation, this maps to weight decay $W\mu$ on adapted parameters and learning rate η/W on excluded biases and normalization parameters. Section 4 tests both cases with native measurements and explicit rewrites.

The stated conditions are essential: varying scales under momentum or zero-norm LARS parameters can break these maps. Proofs and counterexamples (cf. Appendix B) accompany 1,200 paired trajectories with maximum relative difference 8.5×10^{-16}.

3 PARTITIONCHECK

PartitionCheck tests whether two partition plans execute the same logical update. As Figure 2 shows, it first checks that the plans are comparable, then runs the native code under both plans, compares aligned recorded traces under frozen numerical tolerances, and separately checks each execution against available references. It reports *partition consistency* and *reference conformance*, or withholds a verdict for reasons such as *non-comparable*, *unsupported*, *uncalibrated*, or *nonrepeatable*.

3.1 SAME LOGICAL UPDATE, TWO EXECUTIONS

For each test, we specify a logical update $\mathcal{L} = (X_0, \mathcal{A}, \mathcal{C}, T)$ and choose supported partition plans p and q. The validator checks that both plans preserve the initial state, the inputs, the declared contract, and the update boundary, and declines non-comparable and boundary plans (§2.1). A reviewed adapter and support profile then bind the declared computation to the selected framework code and runtime. The adapter executes the native code, not a reimplementation, in fresh processes started from the same initial state. It aligns randomness across the paired runs and records comparable quantities along the update, including objective statistics, selected gradient stages, optimizer state, and parameter updates, where supported; unrecorded stages remain unavailable. Comparisons are always made at the same logical point of the update, and the unit of comparison is the logical update, not an optimizer call: counting optimizer calls alone would miss GaLore's 128 steps per parameter across four declared updates (Table 1).

3.2 COMPARE ALIGNED TRACES UNDER FROZEN TOLERANCES

Exact equality is too strict in floating-point arithmetic, because repartitioning can change the order of operations. PartitionCheck therefore calibrates tolerances on separate control inputs before the formal test and freezes them before comparison (cf. Appendix E.3). It first checks repeatability within each fixed plan; if repeats are unstable, it returns *nonrepeatable* rather than attributing the difference to partitioning. For recorded values x_s, y_s at stage s, we compute $d_s = \|x_s - y_s\|_2 / \max(\|x_s\|_2, \|y_s\|_2, \epsilon_s)$ and $s^\star = \min\{s : d_s > \tau_s\}$, where $\epsilon_s > 0$ is a norm floor and τ_s is the frozen tolerance for that stage. PartitionCheck reports the first recorded stage that repeatedly exceeds its threshold. This *first observed divergence* localizes the discrepancy, but it is not itself a proof of root cause.

3.3 AGREEMENT IS NOT CORRECTNESS

Agreement between p and q does not establish correctness. PartitionCheck therefore reports *reference conformance* separately from partition consistency. For covered operations, the reference is either an FP64 implementation for a small model or a calculation from the recorded inputs and the declared contract. FP64 reduces rounding error but is not exact arithmetic, and operations outside the reference coverage receive no correctness verdict. The separate verdict catches shared errors: in the tested Q-GaLore entry point, seven floating-point parameters never register optimizer hooks, so two executions can agree while both violate a reference that updates all 21 parameters once.

To connect a repeatable discrepancy to a mechanism, we combine code analysis with a targeted repair and a mechanism-removing control. Repair closure requires that the repair remove the first observed discrepancy within tolerance and satisfy the available reference checks. This confirms the mechanism for the chosen partition pair, without establishing its prevalence.

A worked example. In Megatron-Core, two plans group the same 12 ordered non-padding tokens as $p = [4, 4, 4]$ and $q = [6, 2, 4]$; both pass the repeatability check. After two microbatches, the recorded routing fraction is 0.5 under p but 1 under q, and the accumulated gradients differ by 0.25, above the frozen threshold of 1.7×10^{-5}. The reference divides each expert's cumulative token count by the cumulative valid-token count ($4/8 = 0.5$); the code instead uses the number of processed microbatches multiplied by the current microbatch's token count. Using the cumulative count reduces the relative gradient difference to 2.3×10^{-7}, and both repaired executions conform (cf. Appendix A.5).

4 FINDINGS: HOW PARTITIONING REWRITES THE ALGORITHM

We trace each rewrite from code to consequence: we derive the rewritten quantities and measure them in native code (§4.1), implement the rewrite explicitly to test whether it reproduces the affected run (§4.2), measure when it matters (§4.3), and examine published and documented configurations (Appendix F.7).

4.1 PREDICTED, THEN MEASURED

Table 1 compares eight code-derived rewrites with native measurements; all agree. Each follows from the contract and Proposition 2 or the relevant code path. Three test dose responses. Barlow Twins' gradient scales as $1/W$, with measured ratios 1, $\frac{1}{2}$, and $\frac{1}{4}$ on one, two, and four GPUs. Mamba-2's unfused tensor-parallel path uses pG normalization groups, a value fixed before the four-way test. Under fixed padding, SimCSE's auxiliary weights 0.05, 0.1, and 0.2 all execute as zero; restoring the term restores $1{:}2{:}4$ scaling in its loss and gradient (relative residual at most 3.6×10^{-15}). Two measurements test more than overall scaling. Proposition 2 predicts a relative MLM weight of 0.2 for SimCSE on two GPUs; native code gives 0.1999994 in FP32 and 0.2 in FP64. In Barlow Twins, a two-GPU LARS update differs by at most 5.4×10^{-5} from the declared update for adapted weights, but moves biases and normalization parameters only half as far.

Predictions before execution. In five additional codebases, we froze six code-derived predictions and their test criteria before running the corresponding experiments; all six held in controlled CPU

Table 1: Eight predicted rewrites checked in native code. Scale maps use Proposition 2; other rewrites are code-specific (Appendix C). W: data-parallel GPUs; p: tensor-parallel degree; k: accumulation steps; M, M_r: global/local classification-weight sums.

Case / knob	Predicted execution	Native observation
SimCSE / W	Relative MLM weight $W\lambda$	$W = 2$, $\lambda = 0.1$: 0.1999994 (FP32), 0.2 (FP64).
SimCSE family / fixed padding	Auxiliary loss omitted	Weight 0 at $\lambda = 0.05, 0.1, 0.2$; restored term scales 1:2:4.
Qwen3-MoE / W	Auxiliary-loss weight $W\lambda$	At $W = 4$, affected endpoints match repaired 4λ runs in 15/15 archived pairs (post hoc).
Barlow Twins / W	Gradient g/W; adapted weight decay $W\mu$; excluded learning rate η/W	Gradient ratios $1, \frac{1}{2}, \frac{1}{4}$ at $W = 1, 2, 4$; excluded-group updates halve at $W = 2$.
Mamba-2 / p	Normalization groups pG	Declared $G = 4$: 8 groups at $p = 2$; 16 at $p = 4$ (predicted before TP4).
Mask2Former / W	Image weight $M/(WM_r)$	Predicted gradient matches with relative residual at most 1.7×10^{-6}.
GaLore / k	k steps per logical update; clip bypassed	Released 7B setting, $k = 32$: 128 steps/parameter in four updates; no gradient reaches the clip.
veRL KL-Cov / microbatches	At least one selected token per microbatch	Same 8,192 tokens: 1 selected in one microbatch, 8 in eight microbatches.

FP64 tests. FlagEmbedding, GritLM, SLIP, and a separately preregistered Diffusion Policy profile then passed scoped dual-host repair closure, adding four records to the audit; StyleGAN2 did not pass numerical qualification (cf. Appendix F.6).

4.2 Testing the Mapped Run

If the rewrite explains the change, implementing it explicitly should reproduce the affected implementation's recorded behavior. We test this prediction in four cases (Figure 1, bottom; paired differences in Figure F.1). In a new five-seed SimCSE confirmation, the two-GPU run at $\lambda = 0.1$ differs by $+0.43$ STS points (90% CI $[0.09, 0.77]$) from a one-GPU run at the mapped $\lambda = 0.2$ with the loss halved; the interval does not fit inside the preregistered ± 0.5 equivalence margin. Against the declared-weight one-GPU control at the same loss scale, the difference is -3.68 points (95% CI $[-4.56, -2.79]$), passing the new primary test. Mask2Former's affected and mapped implementations agree exactly at every saved checkpoint over 10,000 steps. Barlow Twins' mapped and raw readouts match to recorded precision in all six multi-GPU seed cells. For Mamba-2, an explicit 16-group model on one GPU differs from the affected four-GPU model by $+0.0014$ nats (95% CI $[-0.0010, 0.0038]$) under native 16-group evaluation, inside the preregistered ± 0.005 margin. These tests do not establish full-trajectory identity.

4.3 When the Rewrite Matters—and When It Does Not

The effect of a rewrite on model quality depends on its new value (Table F.1). Mamba-2 at 134M parameters and 100M tokens, trained on the affected four-way path and evaluated as declared, has 0.189 nats more test loss than the repaired model, clearing the preregistered 0.02-nat threshold (perplexity 85.6 against 70.8, 5 seeds). The repaired four-way run matches the one-GPU anchor within the preregistered ± 0.005 margin. The gap (cf. Appendix F.2) comprises 0.136 nats from evaluating 16-group weights with four groups, plus 0.053 nats between models evaluated as trained. In exploratory Barlow Twins runs (ResNet-50, 2,048 steps within warm-up, 3 seeds), raw linear-probe accuracy, below 9% this early, falls with the GPU count by a mean of 1.06 points at $W = 2$ and 1.64 at $W = 4$, while repaired four-GPU runs stay within 0.06 points of one GPU. As Proposition 2 predicts, at $W = 4$ repairing only biases and normalization parameters recovers 1.60 points on average; repairing only adapted weights changes the readout by -0.01 points.

Other rewrites leave the endpoint score almost unchanged. Mask2Former's reweighting changes the class of about 3% of predicted pixels, in 2,497 of 2,500 images, yet moves PQ by less than 0.04 points, far from the prospective 1-point gate. Nor does restoring the declared algorithm always help:

in 60M-parameter LLaMA pre-training it lowers validation loss by 0.21 nats for GaLore and 4.12 for Q-GaLore but raises it by 0.16 for APOLLO. Repair also lowers quality in an exploratory Barlow Twins continuation branched after warm-up and in a preregistered GritLM study, but improves FlagEmbedding's retrieval (cf. Appendices F.3 and F.6).

An ablation that cannot fail. With fixed padding, SimCSE uses a collator that never produces MLM inputs. "MLM on" and "MLM off" agree at every saved checkpoint and evaluation (5 seeds, 2 training corpora); model, buffers, optimizer state, and clipped-gradient summaries match at steps 1, 125, and 15,625, so this ablation yields zero effect by construction. With the objective restored, MLM changes endpoint STS by -13.1 points (95% CI $[-14.7, -11.5]$; secondary) and transfer accuracy by $+0.46$ points (97.5% CI $[0.08, 0.83]$; the preregistered primary, which did not clear its 0.25-point threshold). These directions match the original authors' report (cf. Appendix D in Gao et al., 2021); what the padding flag decides is whether an ablation can see them at all. DiffCSE and the SimCSE++ baseline delete their auxiliary terms through the same dispatch (cf. Appendix C). We cannot assess published ablations without their padding and GPU settings.

5 Audit and Checker Evaluation

Section 4 examined a few rewrites in depth. We now ask how far the instrument behind them can be trusted and how far the findings reach; neither the frozen evaluation roster nor the audit records support rate estimates for code at large.

5.1 How Far the Checker Can Be Trusted

We evaluate PartitionCheck on a frozen roster of 28 rows (Table E.1 in Appendix E): ten injected faults, four real ones, nine benign runs, two repaired arms, and three plans that must be rejected before execution. Every fault is detected and every row classified correctly, and the four shared faults are caught only by reference conformance, as the second verdict intends. The roster is constructed, not sampled: its injected faults are large (the smallest scales a stage by 0.75), its real faults were chosen, and eight of its nine benign runs were held out from calibration. It tests the checker's intended behavior, not recall or false-alarm rates on arbitrary code. A separate 108-row archival replay detects all 36 audited violations, raises no alarm on 36 matched benign configurations, and abstains on all 36 constructed profile-drift probes; it covers mechanism-specific replay, not discovery on unseen code (cf. Appendix E.7). Overhead is small except for one full trace.

5.2 How Far the Findings Reach

Across eight domains, the 31 frameworks and research codebases (Table D.1 in Appendix D) contain 43 violations, each closed by a local repair at the level of one logical update: 37 change with the partition, and 6 are shared by every partition tested. These counts refer to implementation records, not independent discoveries: by responsible code, contract, and repair, the original 39 records form 28 mechanism groups, and all 43 records carry formal, scoped, or case-level validation (cf. Appendix D). We claim no first discovery: six of the 28 groups were publicly reported before our audit, four have related public discussion, and the other 18 carry no novelty claim. Because the audit went looking for mechanisms rather than counting them, a codebase missing from Table D.1 was either not examined or examined without a finding, and neither makes it correct. By 18 September 2026 we had reported violations in 19 of the original 27 codebases through 26 issues and 24 proposed patches, none yet merged. The LightlyTrain and Megatron-Core maintainers have acknowledged reported problems, and an independent contributor reproduced the DeepSpeed AutoEP fault (cf. Appendix D.7); links are withheld for anonymity.

6 Related Work

Checking training implementations. TrainCheck infers execution invariants (Jiang et al., 2025b), TTrace compares executions with reference tensors (Jiang et al., 2025a), OpGuard aligns production runs (Zhou et al., 2026), and TrainVerify proves parallel-plan equivalence to a logical specification (Lu et al., 2025). Other work preserves semantics by construction: shared-prefix reuse preserves a

training schedule in real arithmetic (Li et al., 2026), and Libra rebalances attention without changing model layers or optimizers (Wang et al., 2026). Our focus is complementary: define which complete updates must agree, test whether independently implemented partitions preserve them, distinguish agreement from reference conformance, and derive the algorithms executed when invariance fails.

Determinism, variance, and batch-size invariance. Batch-invariant kernels control floating-point order (He & Thinking Machines Lab, 2025), and variance studies examine seeds, parallelism, and tooling (Pham et al., 2020; Summers & Dinneen, 2021; Zhuang et al., 2022); our violations persist in exact arithmetic. Batch-size-invariant Adam changes how microbatch gradients enter second-moment estimation (Wang & Aitchison, 2024); here the algorithm is fixed, so declared changes are non-comparable, whereas incorrect token-mean accumulation (Lysandre et al., 2024) violates exact merge.

Metamorphic and differential testing. Metamorphic testing uses relations between executions (Chen et al., 2018), CRADLE compares library backends (Pham et al., 2019), and ModelMeta derives model-level relations (Mu et al., 2025). Partition Invariance supplies a training-update relation with explicit admissibility, independent reference checks, and interpretable algorithmic rewrites.

7 Discussion and Limitations

Report the logical update. Report alongside hyperparameters the logical batch in examples and supervised tokens, all parallel degrees, accumulation, padding or packing, and Partition Invariance test settings and verdicts. Frameworks can make this check a regression test; affected users should rerun conclusions that depend on a violation.

What a pass means. A pass means only that no repeatable tested discrepancy exceeded frozen tolerances, not universal invariance or correctness. Calibration uses constructed controls with unestablished exchangeability; seeded faults are large, leaving near-threshold sensitivity unmeasured. Reference coverage and trace visibility are partial, and the first observed divergence is not a root-cause proof.

What the audit can say. The mechanism-seeking audit measures neither prevalence nor discovery speed; six original mechanism groups were already public. Findings depend on audited versions, support domains, and stated contracts where documentation leaves semantics open. Local repair closure is not upstream acceptance.

How far the evidence reaches. Evidence remains bounded by the 134M-parameter, 100M-token Mamba-2 study, a five-seed SimCSE confirmation that did not establish endpoint equivalence, small exploratory Barlow Twins studies, and post-hoc synthetic Qwen3-MoE checks. Proposition 2 requires constant gradient scaling; other rewrites are code-specific, and effects may differ or reverse at scale. Our claims concern executed algorithms and measured quality, not changes to published results, which would require matched reruns of those experiments.

8 Conclusion

Partitioning a training update is treated as an engineering detail. We found that it can change the algorithm, and not at random: a partition rescales a loss coefficient, divides a learning rate, multiplies a group count, or adds optimizer steps, according to the premise of partition invariance that the code breaks. The executed algorithm can therefore be identified, implemented explicitly, and evaluated for its consequences. The price ranged from no detectable change in Mask2Former's panoptic quality to 0.189 nats of test loss in four-way Mamba-2 under declared-architecture evaluation, and restoring the declared algorithm did not always help. The same Barlow Twins hyperparameters, documented at 32 and at 16 GPUs, name two different algorithms under the released code. Partition Invariance turns the neutrality of resources into a property that can be tested; as training spreads across ever more heterogeneous hardware, which algorithm the code executed is worth testing rather than assuming.

AI USE STATEMENT

We used generative AI tools to assist with drafting and revising the manuscript and with writing and editing code. The authors take full responsibility for the final text, code, and scientific claims.

ETHICS STATEMENT

We will report the implementation errors identified in this study to the affected projects through GitHub issues and submit pull requests with proposed fixes. These reports will document the affected versions, reproduction steps, and supporting evidence to help maintainers assess and address each issue.

REPRODUCIBILITY STATEMENT

We will publicly release all code and experimental data produced for this study through a GitHub repository, including experiment configurations, raw results, and scripts for reproducing the tables and figures. The paper provides proofs (cf. Appendix B), case documentation (cf. Appendix D), and environment, seed, and artifact details (cf. Appendix G). Third-party datasets and pretrained checkpoints will be identified by source and version, with instructions for obtaining them under their respective licenses.

REFERENCES

Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, T. H. Tse, and Zhi Quan Zhou. Metamorphic testing: A review of challenges and opportunities. *ACM Computing Surveys*, 51(1):4:1–4:27, 2018. doi: 10.1145/3143561.

Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1290–1299, 2022.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6894–6910, 2021.

Horace He and Thinking Machines Lab. Defeating nondeterminism in LLM inference. Thinking Machines Lab: Connectionism, September 2025. URL https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/.

Haitian Jiang, Shaowei Zhu, Zhen Zhang, Zhenyu Song, Xinwei Fu, Zhen Jia, Yida Wang, and Jinyang Li. TTrace: Lightweight error checking and diagnosis for distributed training. *arXiv preprint arXiv:2506.09280*, 2025a.

Yuxuan Jiang, Ziming Zhou, Boyu Xu, Beijie Liu, Runhui Xu, and Peng Huang. Training with confidence: Catching silent errors in deep learning training with automated proactive checks. *arXiv preprint arXiv:2506.14813*, 2025b.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Pengbo Li, Feiyuan Zhang, Guangming Sheng, Guangxin He, Di Chai, Ziniu Li, Taiqiang Wu, Wenyu Mao, Binhang Yuan, and Kai Chen. Schedule-level shared-prefix reuse for LLM RL training. *arXiv preprint arXiv:2606.01143*, 2026. URL https://arxiv.org/abs/2606.01143.

Yunchi Lu, Youshan Miao, Cheng Tan, Peng Huang, Yi Zhu, Xian Zhang, and Fan Yang. TrainVerify: Equivalence-based verification for distributed LLM training. *arXiv preprint arXiv:2506.15961*, 2025.

Lysandre, Arthur Zucker, Zachary Mueller, Yih-Dar Shieh, Benjamin Bossan, and Pedro Cuenca. Fixing gradient accumulation. Hugging Face Blog, October 2024. URL https://huggingface.co/blog/gradient_accumulation.

Yanzhou Mu, Juan Zhai, Chunrong Fang, Xiang Chen, Zhixiang Cao, Peiran Yang, Kexin Zhao, An Guo, and Zhenyu Chen. Improving deep learning framework testing with model-level metamorphic testing. *Proceedings of the ACM on Software Engineering*, 2(ISSTA), 2025. doi: 10.1145/3728972. URL https://arxiv.org/abs/2507.04354.

Hung Viet Pham, Thibaud Lutellier, Weizhen Qi, and Lin Tan. CRADLE: Cross-backend validation to detect and localize bugs in deep learning libraries. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 1027–1038, 2019. doi: 10.1109/ICSE.2019.00107. URL https://ieeexplore.ieee.org/document/8812095/.

Hung Viet Pham, Shangshu Qian, Jiannan Wang, Thibaud Lutellier, Jonathan Rosenthal, Lin Tan, Yaoliang Yu, and Nachiappan Nagappan. Problems and opportunities in training deep learning software systems: An analysis of variance. In *Proceedings of the 2020 IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2020. URL https://www.cs.purdue.edu/homes/lintan/publications/variance-ase20.pdf.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3505–3506, 2020.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, 2025.

Cecilia Summers and Michael J. Dinneen. Nondeterminism and instability in neural network optimization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9913–9922. PMLR, 2021. URL https://proceedings.mlr.press/v139/summers21a.html.

Xi Wang and Laurence Aitchison. Batch size invariant Adam. *arXiv preprint arXiv:2402.18824*, 2024. URL https://arxiv.org/abs/2402.18824.

Yan Wang, Xiulong Yuan, Kaiming Yang, Jiaxuan Peng, Pengju Lu, Mingzhen Li, Zhipeng Zhang, Chang Si, Zhixiang Ruan, Hongqing Chen, Linlang Jiang, Siyu Wang, Langshi Chen, Rui Men, Man Yuan, Guangming Tan, Yong Li, Weile Jia, and Jingren Zhou. Libra: Taming attention workload skew in long-context LLM training with bounded sequence pool. *arXiv preprint arXiv:2607.23250*, 2026. URL https://arxiv.org/abs/2607.23250v2.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 12310–12320, 2021.

Zhenyu Zhang, Ajay Jaiswal, Lu Yin, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. Q-GaLore: Quantized GaLore with INT4 projection and layer-adaptive low-rank gradients. *arXiv preprint arXiv:2407.08296*, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Ziming Zhou, Yinjie Zhao, Hang Zhu, Wenxiao Wang, Zhihao Bai, Yun Zhang, Shuguang Wang, Haibin Lin, and Peng Huang. OpGuard: Bitwise alignment for precise and general debugging of production LLM training. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, pp. 1385–1406. USENIX Association, 2026.

Donglin Zhuang, Xingyao Zhang, Shuaiwen Leon Song, and Sara Hooker. Randomness in neural network training: Characterizing the impact of tooling. In *Proceedings of Machine Learning and Systems*, volume 4, 2022. URL https://proceedings.mlsys.org/paper_files/paper/2022/hash/427e0e886ebf87538afdf0badb805b7f-Abstract.html.

A Contracts, Admissibility, and Certification

This appendix fixes the vocabulary of §2–§3: what a logical update declares (cf. Appendix A.1), how the validator decides what may be compared (cf. Appendix A.2), how code is executed and traced (cf. Appendix A.3), and how a divergence becomes a validated record (cf. Appendix A.4); Appendix A.5 follows one test from end to end. Figure A.1 shows the flow; it is a conceptual diagram, and actual stages and oracle coverage depend on the adapter.

A.1 The logical-update declaration

A logical update and each candidate plan are serialized objects with closed field sets (Table A.1). Unknown fields are rejected; every identifier is the SHA-256 of canonical UTF-8 JSON with sorted keys, finite numbers, and one trailing newline, so insertion order cannot change an identity. The tensor-state digest (parameters and persistent buffers) and the model-semantics digest (forward implementation, admitted configuration, training mode, adapter and source revisions) are kept separate. The declared transition T may be implemented by several function calls or buffer writes; Proposition 1 covers updates with a deterministic transition after all required statistics have been combined. The table lists the frozen version 2 declaration used by the validator and its regression tests. The exact adapters of §5 do not execute from this generic declaration: each binds its case's declarations in a reviewed support profile (cf. Appendix A.3).

A.2 Admissibility

The validator compares a plan's declarations with the update's, never with a run, and returns one of three statuses. A plan is *non-comparable* if it changes an identity or a declared semantic: the update identity; the atom multiset, in the manifest or in the flattened microbatch plan; any atom field; rank coverage (each atom on exactly one rank); the initial-state digests; the model semantics; the policy version; the boundary, transition count, optimizer, or hyperparameters; the contract version; the objective, its normalization, or its reduction; identity-keyed randomness; or the clipping and numeric declarations. A plan is a *boundary* if its declarations leave a batch-coupled operator without its required global information (reason code `global_information_not_preserved`) or declare global clipping without global statistics or before the global reduction (`clipping_global_information_missing`). A non-comparable finding dominates; a boundary is reported only when nothing is non-comparable; otherwise the plan is *admissible*. Each finding names the rule, the atom if any, and the expected and actual values.

Concretely, reordering atoms, regrouping them into microbatches or packed sequences, accumulating over more or fewer microbatches, moving atoms between ranks, and changing sharding or reduction order within the declared numeric mode are all admissible. Dropping, duplicating, or padding an atom into the loss, keying randomness by rank, declaring a different clipping scope, declaring rank-local negatives where the update declares global ones, and declaring a second transition are non-comparable. Declaring an in-batch-negative or batch-normalization objective without preserved global information is a boundary. None of these labels is a verdict: a violation is diagnosed only after an admissible plan runs and its trace leaves the calibrated envelope or disagrees with the oracle.

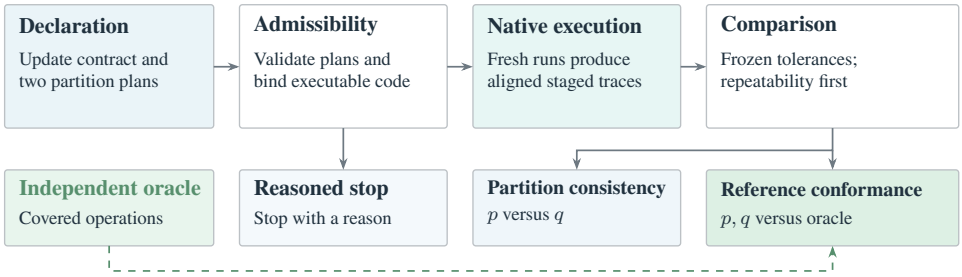

Figure A.1: Checker flow (conceptual). A test takes a declared update and two plans; admissibility is decided before code execution, execution produces staged traces, and the two verdicts are reported separately. Invalid, unsupported, uncalibrated, and nonrepeatable cases stop with a reason.

A.3 Exact adapters, qualification, and the staged trace

An exact adapter runs the bound code, not a reimplementation. It is described by an immutable descriptor for one revision and one support profile: the adapter identity; a revision digest derived from the reviewed implementation, the dependency lock, the profile digests, and the conformance suite, so that any change to source or dependencies changes the revision; a lifecycle state; the profile identity; the stages it emits; and the oracles it supports. The registry that resolves descriptors is closed, with no plugin lookup, default revision, or fallback, and it recomputes each descriptor digest on every resolution. Qualification runs in a fresh, isolated interpreter; any mismatch of identity, construction, or runtime against the reviewed profile returns *unsupported* before the first forward pass.

The staged trace records, in causal order: objective numerator, objective mass, objective coefficients, local gradient, unscaled gradient, reduced gradient, pre-clip norm, clip coefficient, post-clip gradient, optimizer step counter, optimizer-state summary, and parameter delta. A real adapter may cover a prefix or a subset. Profiles with an additional absolute criterion require both the relative and the absolute threshold to be exceeded. Stages it does not cover are unavailable to both verdicts, and a first divergent stage counts as localized only if every earlier stage was recorded (cf. Appendix E).

A.4 From a divergence to a validated record

A violation becomes a record only after repair closure (§3): the divergence must repeat, a local change to the implicated operation must remove it and agree with the oracle, a control without the suspected mechanism must show nothing, and, where the protocol asks for it, a second host must reproduce the whole. Records reach this bar with different levels of evidence, which Table D.2 reports per record:

- *Scientific*: repair, negative control, and oracle agreement are recorded in a fixed support domain. *Scoped* records restrict that domain further, for example to FP32 or FP64 execution or to a small model; *supporting* records share their mechanism with another record and add no mechanism of their own; *reviewed* records carry a recorded author review of the classification and support domain.
- *Formal, scoped*: the record passed a formal qualification chain within an accepted support domain.
- *Case-level*: the record is confirmed within an explicitly reviewed contract.
- *Technical closure only*: automated technical and semantic adjudication in a fixed one-update domain, with no recorded author review.

All 39 records are now at one of the first three levels; the three formerly technical-only records passed scoped follow-up checks (cf. Appendix E.7). None of these levels asserts correctness beyond its support domain.

Table A.1: Fields of the logical-update declaration (contract version 2), as frozen in the validator's schema.

Object	Fields	Role
`AtomSpec`	atom identity, weight, mass, supervision and label masks, position identifiers, policy and data versions, view and augmentation identity, payload digest	identity and metadata of one training item
`ObjectiveContract`	objective kind, normalization, loss reduction, batch-coupled operator, required global information	the scalar objective the update optimizes
`RNGContract`	mode, global seed, identity fields, algorithm	randomness keyed to identities, not to ranks or microbatch positions
`ClippingContract`	mode, threshold, application point, global statistics	whether and where clipping applies
`NumericContract`	numeric mode, dtype, tolerances, reduction order	reference versus production semantics
`ContractSpec`	contract version with the four contracts above	the versioned algorithm declaration
`ModelSemanticSpec`	forward digest, configuration digest, training mode, adapter identity and revision, source revision	executable model identity, separate from tensor state
`LogicalUpdateSpec`	update identity, atoms, initial-state digests, model semantics, boundary, policy version, contract, optimizer, transition count	the fixed update under test
`PartitionSpec`	realized atoms, microbatches and rank assignment, state digests, model semantics, boundary, policy version, declared contract, transition count	one concrete plan submitted for admissibility

A.5 A WORKED EXAMPLE: THE MEGATRON-CORE ROUTING DENOMINATOR

This example follows one test through the checker of §3; the case and its support domain appear in Appendix C. Both executions process the same 12 ordered non-padding tokens, but group them as $p = [4, 4, 4]$ and $q = [6, 2, 4]$. Under the constructed repeated inputs, this repartition isolates the routing denominator while preserving the intended prefix-based computation.

Both executions pass the repeatability check. After two microbatches, each has processed eight valid tokens, four per selected expert. The recorded routing fraction is 0.5 under p but 1 under q, and the final accumulated gradients differ by 0.25, above the frozen threshold of 1.7×10^{-5} (cf. Appendix E.3).

The reference divides each expert's cumulative token count by the cumulative valid-token count, giving $4/8 = 0.5$ after the second microbatch. The original code instead uses the number of processed microbatches multiplied by the current microbatch's token count, which yields 0.5 under p and 1 under q. Replacing this denominator with the cumulative count reduces the relative gradient difference from 0.25 to 2.3×10^{-7}, below the frozen threshold, and both repaired executions conform. The first observed divergence, the reference disagreement, and the repair together establish repair closure for this partition pair (§3.3).

B PROOFS AND BOUNDARY CASES

B.1 PROOF OF PROPOSITION 1

Fix $\mathcal{L} = (X_0, \mathcal{A}, \mathcal{C}, T)$ and two admissible partitions $p = (\mathcal{A}_1, \ldots, \mathcal{A}_m)$ and $q = (\mathcal{B}_1, \ldots, \mathcal{B}_n)$. Both are disjoint regroupings of the same identity-bearing multiset, because admissibility preserves every atom and premise (i) preserves every atom field. Process the stages in causal order. For stage

j, associativity and commutativity of $\oplus_j$ give

$$S_j(\mathcal{A}_1) \oplus_j \cdots \oplus_j S_j(\mathcal{A}_m) = S_j(\mathcal{A}) = S_j(\mathcal{B}_1) \oplus_j \cdots \oplus_j S_j(\mathcal{B}_n),$$

so both partitions form the same merged statistic, provided each block's statistic is computed from the same inputs. That holds inductively: random draws are the same functions of seed, update, atom, operation, and view under both partitions (premise ii); every batch-coupled stage sees the global information its contract requires rather than a partition-local substitute (premise iii); and each stage value is a deterministic function of X_0, the merged statistics of earlier stages, and $\mathcal{C}$ (premise iv). Hence every stage value agrees, and because exactly one deterministic transition follows all merges (premise v), so does the final state. The condition is sufficient, not necessary: an implementation can satisfy Partition Invariance through an equivalence the premises do not capture. An executable fixture checks the merge identity on 100 random partitions with maximum absolute error 1.1×10^{-13} (seed 1729).

B.2 What admissibility excludes

Proposition B.1 (Segmented transitions)**.** Replacing the single declared transition by several intermediate transitions need not preserve the declared update.

Witness. Take $\ell_i(\theta) = \frac{1}{2}(\theta - a_i)^2$ for $i = 1, 2$, initial parameter θ_0, and learning rate η. One SGD transition on the mean gradient gives $\theta_{\text{one}} = \theta_0 - \frac{\eta}{2}\big[(\theta_0 - a_1) + (\theta_0 - a_2)\big]$. Two transitions of rate $\eta/2$, one per atom, give $\theta_1 = \theta_0 - \frac{\eta}{2}(\theta_0 - a_1)$ and $\theta_{\text{seg}} = \theta_1 - \frac{\eta}{2}(\theta_1 - a_2)$, so $\theta_{\text{seg}} - \theta_{\text{one}} = \frac{\eta^2}{4}(\theta_0 - a_1)$. The second gradient is evaluated at moved parameters, although the atoms and the total learning rate are unchanged. Equality holds in degenerate cases, namely $\eta = 0$, $\theta_0 = a_1$ for this ordered witness, losses with constant gradients, and other commuting transitions, so the statement is "not in general" rather than "never". The executable fixture verifies the identity symbolically, to 80 digits, and finds a nonzero difference in all 100 predeclared nondegenerate samples. Consequently a plan that declares extra transitions is non-comparable, while an implementation that commits state more than once per declared update violates premise (v) of Proposition 1.

Proposition B.2 (Missing global information)**.** Suppose the declared output of a batch-coupled operator depends on a global statistic, and a rank-local view does not determine that statistic. Then no function of the rank-local view reproduces the declared operator on all inputs.

Proof and witness. If the local view does not determine the statistic, two admissible global inputs share the same local view but require different outputs; any function of the view returns the same value on both and cannot match both. In the fixture, one positive query–document pair sits on each of two ranks, the declared objective uses the full 2×2 logit matrix, and the logits are dot products with one scalar parameter. The rank-local 1×1 softmaxes have loss and gradient 0, whereas the declared objective has loss 1.5397 and gradient 1.6875; a differentiable gather restores the full matrix and matches both values exactly. A plan without the required statistic is therefore a boundary, not a fault. Restoring it, by synchronized normalization, a differentiable gather, or globally merged statistics, returns the operator to the scope of Proposition 1.

B.3 Proof of Proposition 2

Assumptions. Two executions share initial parameters, update schedule, inputs, and random identities. At every common parameter value and update t, the affected data gradient is $\widetilde{g}_t = c\,h_t$, where h_t is the mapped execution's data gradient and $c > 0$ is fixed for the whole trajectory. Arithmetic is exact. Momentum and Adam moments start at zero or are transformed as stated. Learning-rate schedules are prescribed and mapped pointwise; data-dependent schedulers, skipped steps, state-dependent scaling, and unspecified fused variants are outside the statement. Weight decay is taken at its actual position in the implementation. The proof is by induction on t: equal parameters at step t make the gradient relation applicable, and the step preserves equality.

Heavy-ball SGD. With coupled decay μ, the affected momentum input is $u_t = c\,h_t + \mu\theta_t = c\,(h_t + (\mu/c)\theta_t) = c\,u'_t$. For $v_t = \beta v_{t-1} + u_t$ and $v'_t = \beta v'_{t-1} + u'_t$, zero initial states give $v_t = c\,v'_t$, so $-\eta_t v_t = -(c\,\eta_t)\,v'_t$. The map is $\eta_t \mapsto c\,\eta_t$ and $\mu \mapsto \mu/c$; mapping the learning rate alone is not enough when coupled decay is present. A decoupled multiplicative decay would instead require preserving the product $\eta_t\mu_t$ and is not covered.

Adam and AdamW. With equal β_1, β_2 and step counters, the recurrences give $m_t = c\,m'_t$ and $v_t = c^2 v'_t$, and bias correction preserves both relations. Because $c > 0$,

$$\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} = \frac{c\,\hat{m}'_t}{c\sqrt{\hat{v}'_t} + \epsilon} = \frac{\hat{m}'_t}{\sqrt{\hat{v}'_t} + \epsilon/c}.$$

Coupled regularization needs $\mu \mapsto \mu/c$ before the moment recurrences. For AdamW the learning rate is unchanged, so the decoupled decay is identical with $\mu' = \mu$. The argument depends on ϵ sitting outside the square root; other placements need their own rule.

LARS as implemented in Barlow Twins. For an adapted group, $u = c\,h + \mu\theta = c\,(h + (\mu/c)\theta) = c\,u'$. With trust coefficient α and nonzero θ and u,

$$\frac{\alpha\|\theta\|}{\|u\|}\,u = \frac{\alpha\|\theta\|}{c\|u'\|}\,c\,u' = \frac{\alpha\|\theta\|}{\|u'\|}\,u',$$

so the vectors entering momentum are identical and the map is $\mu \mapsto \mu/c$ with the learning rate unchanged. If $u = 0$, both vectors vanish and the implementation's unit fallback is harmless. If $\theta = 0$ but $u \neq 0$, the implementation uses a unit trust ratio, the inputs differ by c, and the decay-only map fails; this branch is excluded. Biases and normalization parameters are excluded from adaptation and decay in the pinned source, so their recurrence is heavy-ball SGD and needs $\eta_t \mapsto c\,\eta_t$. The derivation covers only this implementation, which has no additive stabilizer in the trust ratio.

Clipping. For ideal global-norm clipping $C_\tau(g) = g\min(1, \tau/\|g\|)$, defined as 0 at $g = 0$, homogeneity gives $C_\tau(c\,h) = c\,C_{\tau/c}(h)$, which composes with the optimizer maps above when applied at the same point in both pipelines. With an additive stabilizer, $C_{\tau,\kappa}(g) = g\min(1, \tau/(\|g\| + \kappa))$, both τ and κ must be divided by c. The installed PyTorch clip uses a fixed $\kappa = 10^{-6}$, so a threshold-only map is not an exact identity for native clipping.

Unequal term scales. For $\widetilde{g}_t = \sum_k a_k \nabla \ell_k$ and any fixed $c > 0$, setting $\lambda'_k = a_k/c$ gives $\widetilde{g}_t = c\sum_k \lambda'_k \nabla \ell_k$, and the maps above apply to the mapped objective. The identity does not imply that the objective changes. If the component gradients are linearly independent at a state, non-proportional coefficient vectors give total gradients that are not scalar multiples; non-collinearity alone is weaker. For $\ell_1(x, y) = x$, $\ell_2(x, y) = y$, and $\ell_3(x, y) = x + y$, the gradients are not all collinear, yet the coefficients $(1, 1, 1)$ and $(\frac{1}{2}, \frac{1}{2}, \frac{3}{2})$ define the same objective, $2x + 2y$. Nor do different gradient directions imply different updates for every optimizer: from zero moments with $\epsilon = 0$, Adam maps the first-step gradients $(1, 1)$ and $(1, 2)$ to the same normalized update.

Optimizer states. The maps match parameter sequences, not optimizer states. Adam's first and second moments scale by c and c^2, heavy-ball momentum by c, while momentum in LARS's adapted groups is unchanged.

B.4 Numerical check and counterexamples

The maps were checked on 1,200 paired trajectories: 12 settings of 100 random trials each, 12 updates per trajectory, c drawn from $[1/8, 8]$, FP64 on CPU with the native PyTorch 2.13 optimizers (foreach and fused paths disabled) and the LARS class extracted unmodified from the Barlow Twins repository at commit `8e8d284c`. The relative difference is $\|x - y\|_2 / \max(\|x\|_2, \|y\|_2, 10^{-300})$, with updates computed from native optimizer states; the acceptance tolerance, fixed in advance, is 10^{-12}. All 1,200 trajectories pass, with a largest relative difference of 8.5×10^{-16} (Table B.1, Figure B.1). These are numerical witnesses in a CPU fixture, not independent training runs, and they do not replace the proof. Table B.2 records the ten boundary and control observations; each behaves as the assumptions predict.

Table B.1: Largest relative differences over 100 trajectories per setting.

Setting	Parameters	Updates	Optimizer state
SGD	3.8×10^{-16}	3.8×10^{-16}	3.7×10^{-16}
SGD, coupled decay	4.2×10^{-16}	3.6×10^{-16}	3.7×10^{-16}
Adam	1.8×10^{-16}	3.8×10^{-16}	3.7×10^{-16}
Adam, coupled decay	1.9×10^{-16}	4.1×10^{-16}	4.4×10^{-16}
AdamW	1.6×10^{-16}	3.7×10^{-16}	3.8×10^{-16}
LARS, adapted groups	9.9×10^{-17}	4.3×10^{-16}	4.2×10^{-16}
LARS, excluded groups	3.8×10^{-16}	3.8×10^{-16}	3.7×10^{-16}
Ideal clipping, SGD	2.3×10^{-16}	4.1×10^{-16}	4.2×10^{-16}
Ideal clipping, Adam	1.6×10^{-16}	4.3×10^{-16}	8.5×10^{-16}
Unequal terms, SGD	3.4×10^{-16}	4.0×10^{-16}	3.9×10^{-16}
Unequal terms, Adam	1.3×10^{-16}	5.0×10^{-16}	4.6×10^{-16}
Unequal terms, AdamW	1.5×10^{-16}	5.1×10^{-16}	4.6×10^{-16}

Table B.2: Boundary and control observations for Proposition 2. "Differs" means the stated map does not reproduce the trajectory, as the assumptions predict.

Observation	Relative difference	Outcome
LARS parameter with zero norm, decay-only map	0.5	differs
Native clipping with a fixed stabilizer, threshold-only map	1.0×10^{-7}	differs
Clipping with threshold and stabilizer both mapped	0	equal
Linearly dependent components, different coefficients	0	equal
Adam at $\epsilon = 0$, first step, different directions	0	equal
Scale c varying over time, with momentum	0.114	differs
Nonzero initial momentum left unmapped	0.122	differs
SGD with coupled decay, learning rate mapped alone	0.0127	differs
Two unequal terms, common scale mapped only	0.279	differs
Two unequal terms, coefficient map applied	0	equal

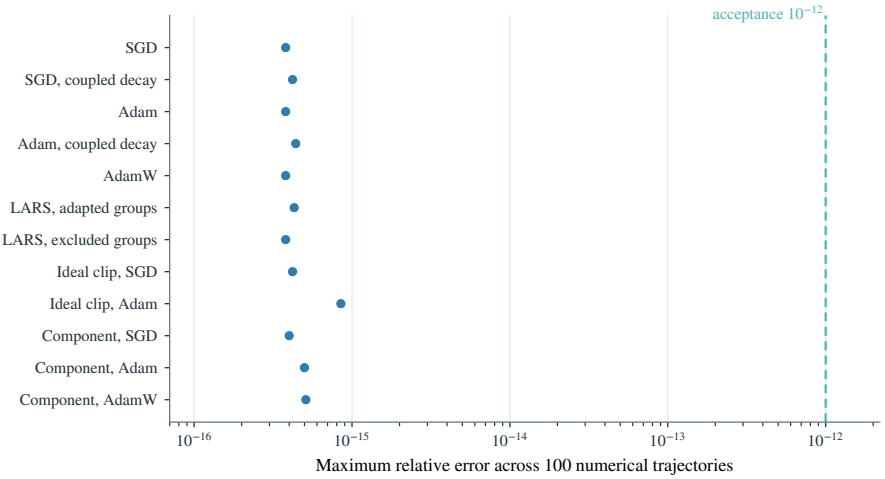

Figure B.1: Largest relative error of each mapping over 100 numerical trajectories, against the acceptance threshold 10^{-12} (dashed). CPU fixture; not a substitute for the proof.

C Case Studies: Explaining and Validating Violations

Each case below gives the mechanism, the derived computation and its source (Proposition 2 or the code path), the measurement in native code, the evidence that closes it (repair, oracle, negative control, second host), and the limits of its support domain. Unless stated otherwise, a derivation was made from the code; we say that a value was fixed before measurement only where a record documents it. Distances are relative L_2 distances between the two partitions at the named stage, before and after the local repair. Records, mechanism groups, and validation levels follow Table D.2; the cases are reading groups and add no counts.

C.1 SimCSE and DiffCSE: the auxiliary weight under data parallelism

Mechanism. The contrastive loss is computed on a gathered batch whose non-local slices are detached, so each GPU backpropagates only through its own slice; data parallelism then averages. The auxiliary MLM (SimCSE) or RTD (DiffCSE) term is local and averages correctly. *Derivation.* The contrastive term receives $a = 1/W$ and the auxiliary term λ; with $c = 1/W$, Proposition 2 gives $\lambda' = W\lambda$, and AdamW absorbs the common factor through ϵ. *Measurement.* At $W = 2$ and $\lambda = 0.1$ the executed relative weight is 0.1999994344 in FP32 and exactly 0.2 in FP64; the two-GPU gradient decomposes as $\frac{1}{2}g_{\mathrm{con}} + \lambda g_{\mathrm{aux}}$, and a component repair restores $g_{\mathrm{con}} + \lambda g_{\mathrm{aux}}$. *Closure.* Two independent hosts with two GPUs each, eight profiles, 360 complete native single steps, and three technical repeats of the formal seed in AB/BA/AB order; held-out benign comparisons raised no alarm (0/16), which we do not report as a population rate. The mapped-run prediction ($\lambda = 0.2$ with the loss halved on one GPU) is written into the task-study protocol (cf. Appendix F). *Scope.* Fixed global batch of 64, identity-keyed views, equal auxiliary supervision mass, FP32 and FP64.

C.2 The SimCSE family under fixed padding

Mechanism. With padding fixed, the training script routes batches through a collator that never produces the auxiliary inputs, so the MLM (SimCSE, SimCSE++ baseline) or RTD (DiffCSE) term is never computed although its flag is on. *Derivation.* From the dispatch code: the executed auxiliary weight is 0 for every declared λ. *Measurement.* For $\lambda \in \{0.05, 0.1, 0.2\}$ the affected path produces no auxiliary loss and no encoder auxiliary gradient; with the collator repaired, the loss and gradient norm scale $1{:}2{:}4$, with relative residual at most 3.6×10^{-15} (CPU, FP64, fixed batch, single-step checks). In full training, "MLM on" coincides with "MLM off" at every saved checkpoint (steps 1, 125, and 15,625; model, buffers, optimizer state, and clipped-gradient summaries; 60 state comparisons) and in every evaluation score. For DiffCSE, the affected run is identical to an explicit run with RTD off in the loss, all 77 parameter gradients, model and buffers, optimizer and scheduler state, and random state. The SimCSE++ baseline script is byte-identical to one version of SimCSE's. *Scope.* The formal domain is a small model (hidden size 32, one layer, batch of 4 with two views, dropout off) in FP32 and FP64; we make no claim about the settings of any published ablation.

C.3 Qwen3-MoE in Transformers: the router auxiliary loss

Mechanism. Data-parallel world-size compensation is applied to a router auxiliary statistic computed rank-locally, which rescales the auxiliary term alone. *Derivation.* By Proposition 2, the auxiliary coefficient becomes $W\lambda$. *Measurement.* On one GPU versus two with the same global data and a mirrored local distribution, the router gradient differs by 43.62% and by 3.4×10^{-7} after repair (Transformers 5.15.0). *Endpoint check.* In archived four-GPU runs of 160 updates on a synthetic task, the affected run at λ and the repaired run at 4λ have identical held-out scores and recorded final-parameter hashes in all 15 pairs (five synthetic task instances, three coefficient pairs). This is a post-hoc comparison of archived records, not a new trajectory-level equivalence. *Scope.* The formal case is one of the three release lanes (cf. Appendix E).

C.4 Megatron-Core: the router token mass

Mechanism. Under the approximate-prefix buffer, the auxiliary loss divides by the token mass of the current microbatch rather than by the mass the declared statistic requires, so splitting 12 atoms as 4/4/4 or 6/2/4 gives different updates. *Measurement.* The gradient differs by 25.0% and by

2.3×10^{-7} after repair (Megatron-Core 0.18.2). *Scope.* Approximate-prefix versus exact-global computation is itself a boundary; the qualified pair lies within the approximate path. This is a release lane (cf. Appendix E).

C.5 Barlow Twins and solo-learn

Mechanism. The official implementation builds the cross-correlation matrix through an all-reduce that is not differentiable, so gradients flow only through each GPU's own share before data parallelism averages them, and every term receives $c = 1/W$. *Derivation.* By Proposition 2 for this LARS implementation, adapted groups execute weight decay $W\mu$ and the excluded bias and normalization groups execute learning rate η/W. *Measurement.* On one versus two GPUs the gradient differs by 0.49999996 and the update by 0.49965797 on both hosts; the first divergence is at the gradients. Within the update, adapted weights move within 5.4×10^{-5} of the declared update while the excluded groups differ by 0.49999995. Within-layout mapping checks from zero state and from states reached after 1, 128, and 1,024 steps, continued for 16 steps, pass; detailed per-state values are archived for one seed. *Scope.* Scoped FP32 formal closure for one update on two hosts (cf. Appendix E.7); separate two- and four-GPU independent global-reference checks pass. The three-seed short training study remains exploratory. solo-learn is a separate mechanism: a batch normalization inside the loss, closed in FP64 (update distance 0.587 before and 2.7×10^{-14} after a joint repair).

C.6 Mamba-2: normalization groups under tensor parallelism

Mechanism. On the unfused tensor-parallel path, each shard normalizes over its own slice with the declared group width, so the global number of normalization groups becomes pG under contiguous sharding and divisibility; the B/C grouping is unchanged. *Measurement.* With $G = 4$, the path executes 4, 8, and 16 groups at $p = 1, 2, 4$. The four-way value was fixed before any four-way run, and three four-way checks passed. Native two-way forward passes of 15 trained checkpoints match the explicit eight-group model with relative output residual at most 4.1×10^{-5}, and gradients for three of them at most 8.5×10^{-5}. *Checkpoint semantics.* Reloading an affected two-way checkpoint on one GPU under the declared grouping raises its test NLL from 4.28256783 to 4.33697785 nats; with the executed grouping recorded in the configuration, the reload gives 4.28256781, and all 3,270 parameter tensors are byte-identical across the three loads. The grouping metadata was reconstructed from the sealed configuration and a native replay, not recorded at training time. *Scope.* Scoped FP32 closure on a small native module; the mechanism was publicly reported before our audit.

C.7 Mask2Former and OneFormer: per-image classification weights

Mechanism. The weighted cross-entropy is averaged over each rank's images, so image r carries weight $\omega_r = M/(WM_r)$ relative to the declared global weighted mean; mask and Dice terms are unaffected. *Measurement.* Full-gradient predictions from this formula match native gradients with relative residual at most 1.7×10^{-6}; at shared states after 2,000 and 5,000 steps the reweighting changes the full gradient by 3.5–15.5% and the actual update by 2.3–6.9%. In natural COCO batches, the implicit weight is off by more than 1% in 96.9% and 97.9% of 1,024 image occurrences at two and four partitions, a formula statistic on CPU rather than a count of independent images. A one-GPU run replaying the predicted weights matches the two-GPU run's parameters, optimizer state, buffers, and PQ exactly at 2k, 5k, and 10k steps for both seeds. *Scope.* Mask2Former now passes scoped dual-host formal confirmation, with its earlier failures and comparator amendment documented (cf. Appendix E.7); OneFormer, the same mechanism, passed a formal scoped qualification (1,030 of 1,030 checks on two hosts). Both conclusions hold for the audited global weighted-mean contract.

C.8 GaLore: the update clock and the outer clip

Mechanism. Per-layer optimizer hooks consume each parameter's gradient and step inside every backward pass, so accumulating k microbatches executes k transitions per declared update, and the outer gradient clip finds no gradients. *Measurement.* For the README configuration (7B model, 6.738B parameters, microbatch 16, accumulation to 512, sequence length 256, clipping

1.0), four logical updates execute 128 transitions per parameter and the outer clip consumes no gradient; a 32-update natural window executes 1,024. The repair clips and steps once at the logical boundary. *Resources.* The 501-update memory windows and the four-update test on a physical 24 GB card are reported separately (cf. Appendix F). *Scope.* This is a native audit, separate from the scoped qualification of the same clock mechanism on a small model; the accumulation behavior was discussed publicly before our audit.

C.9 Q-GaLore: an omitted hook and the continuation clock

Mechanism. The per-parameter entry sets a plain attribute where an autograd hook is needed, so floating-point parameters receive gradients but are never stepped. *Measurement.* With one accumulation step (FP32, one CUDA rank, a two-layer 128-dimensional native LLaMA, vocabulary 512, rank 32), the 14 quantized weights step once and 7 floating-point parameters (embedding, output head, five normalization weights) never do. Registering the post-accumulate hook makes all 21 parameters step once and matches an independent full-backward oracle bit for bit in model, dequantized parameters, quantization metadata, optimizer, and scheduler on two hosts; the oracle schedules updates independently but shares the native optimizer kernels. *Continuation.* Resuming from cached projections, the GaLore clock mechanism reappears; it is a scoped record of the same group as GaLore's.

C.10 APOLLO: the per-parameter scheduler

Mechanism. In two per-layer entry points, even with one accumulation step, the scheduler counts in the wrong units, doubling warm-up and horizon: the executed learning rates are 0.00125 and 0.0025 where the reference gives 0.0025 and 0.005, and the updates differ by about 0.5. *Closure.* The repair restores the full state exactly. *Consequence.* Restoring the declared scheduler raises validation loss (cf. Appendix F); conformance and quality are different questions. *Scope.* Scoped shared violation; the two entry points are one mechanism.

C.11 veRL KL-Cov: token selection per microbatch

Mechanism. Native FSDP dynamic microbatching recomputes the centered covariance and the minimum-one top-k quota inside each loss invocation. *Measurement.* The same update as one microbatch of 8,192 tokens selects 1 token; as eight of 1,024 it selects 8. The parameter update differs by 7.0×10^{-3}, and by 3.5×10^{-6} after a global-mask repair. *Scope.* A single-update mechanism on the KL-Cov loss path; no consequence for full RLVR training is claimed.

C.12 LightlyTrain DINO, DenseCL, and TRL BCO: auxiliary-state clocks

Mechanism. Under gradient accumulation, the DINO teacher and center (and, in the same group, DenseCL's key EMA and queue and BCO's running moments) are committed per microbatch rather than once per declared update. *Measurement.* DINO: the center differs by 0.694 and the post-clip gradient by 7.4×10^{-4}, falling to 0 and 5.9×10^{-8} after repair (LightlyTrain 0.17.0). DenseCL: the gradient differs by 0.673 and by 1.5×10^{-7} after repair. BCO: the parameter update differs by 0.067, and the repaired residual is exactly zero (TRL 1.12.0). *Scope.* We do not state a closed form for the teacher momentum, because the batch statistics vary between commits.

C.13 bitsandbytes and TorchAO: 8-bit state eligibility

Mechanism. Eligibility for 8-bit optimizer state is decided on the local flattened or sharded shape rather than on the original parameter, so a legal change of wrapping or mesh changes which parameters get low-precision state. *Measurement.* bitsandbytes under FSDP1: the optimizer state differs by relative distance 1.09 (maximum absolute difference 100) and the second update's projection by 3.0×10^{-4}; a repair keyed to original parameter names, an FP32 negative control, and an independent oracle close it exactly, on an independent host as well. TorchAO under FSDP2 shows the same mechanism (state distance 0.75, update projection 2.9×10^{-4}), closed exactly under its corrected lineage. *Scope.* Our result holds for the explicit eligibility policy of the audited integra-

tion; the documentation specifies a size threshold but not how sharded parameters keep their original identity.

C.14 ADAM-MINI: BLOCK STATISTICS

Mechanism. Under FSDP1 with original parameters, the number of visible GPUs is used as the communication group size, so a default scalar block's statistics are not reduced across ranks. *Scope.* Scoped scientific closure with recorded review.

D FULL AUDIT AND COUNTING RULES

D.1 AUDIT BY DOMAIN

D.2 VIOLATION INVENTORY

Table D.2 lists the original 39 implementation records of §5; four prospective additions are listed below the table. Each row gives the audited codebase, the violated mechanism, its mechanism group and the class of that group in Table D.5 (R: reduction scale; L: local statistic; O: operator, precision, or dispatch; C: clock), the validation level (cf. Appendix A.4), the public-knownness mark of its group (cf. Appendix D.5), and the audited version or commit as recorded. Key measurements are provided for the case groups studied in detail (cf. Appendix C).

Contract sources. Each record is judged against an audited contract: the declared objective and normalization, the documented semantics of the operation, or an explicitly stated audit convention. The record-level source bindings below identify these bases and their scope. Versions in Table D.2 come from the frozen audit records; they do not describe current upstream status.

Table D.2: The original 39 implementation records. [s]Shared by every partition tested. [†]Group publicly reported before our audit; [‡]related public discussion. Validation levels are defined separately (cf. Appendix A.4); "reviewed" means an author review of the classification and support domain is recorded.

ID	Codebase	Violation	Group	Validation	Version
I01	Transformers	Qwen3-MoE auxiliary-loss scale	M01 (R)	scientific	5.15.0 (`5eddc12`)
I02	Megatron-Core	router token mass	M02 (L)	scientific	0.18.2 (`571370c`)
I03	OpenCLIP	NaFlex window mass	M03 (L)	scientific, supporting	`602d4af`
I04	Accelerate	fused-AMP skipped steps	M04 (C)	scientific	1.14.0
I05	DeepSpeed	AutoEP clipping norm	M05 (L)	scientific	`cf443004`
I06	LightlyTrain	DINO teacher and center	M06 (C)	scientific	0.17.0
I07	veRL	KL-Cov selection	M07 (L)	scientific	`24f25b03`
I08	bitsandbytes	8-bit state eligibility, FSDP1	M08 (O)	scientific	0.50.2
I09	NeMo RL + Automodel	dynamic CE–KD weights	M09 (L)	scientific, reviewed	`fd45cb8e`, `1814c6c9`
I10	Canzona	SOAP owner state	M10 (O)	scientific, reviewed	`426e8c50`
I11	Canzona	Muon owner state	M10 (O)	scientific, reviewed	`426e8c50`
I12	Adam-mini	block statistics	M11 (O)	scientific, reviewed	‡ `402aadfe`
I13	Q-GaLore	floating-point parameter hook[s]	M12 (O)	scientific, reviewed	`5bd2f1c5`
I14	LightlyTrain	DenseCL key EMA and queue	M06 (C)	scientific, supporting	0.17.0
I15	veRL	GMPO sequence means	M13 (L)	scientific, supporting	† `24f25b03`
I16	TorchAO	8-bit state eligibility, FSDP2	M08 (O)	scientific, supporting	`4e42c02f`
I17	TRL	BCO running moments	M06 (C)	scientific, supporting	1.12.0
I18	NeMo RL + Automodel	multi-teacher selection	M09 (L)	scientific, supporting, reviewed	`fd45cb8e`, `1814c6c9`
I19	Megatron-Core	TP label smoothing	M14 (L)	scientific, reviewed	† `d4550898`
I20	AReaL	M2PO selection and mass	M15 (L)	scientific, reviewed	† `f289b989`
I21	GaLore	per-layer update clock	M16 (C)	scientific, scoped, reviewed	† `2cc66f88`
I22	Q-GaLore	continuation update clock	M16 (C)	scientific, scoped, reviewed	† `5bd2f1c5`
I23	APOLLO	scheduler, per-layer entry[s]	M17 (C)	scientific, scoped, reviewed	`830f7646`
I24	APOLLO	scheduler, 8-bit per-layer entry[s]	M17 (C)	scientific, scoped, reviewed	`830f7646`
I25	TRL	GMPO with Liger dispatch[s]	M18 (O)	scientific, scoped, reviewed	† `32fc9d71`
I26	SimCSE	fixed-padding MLM dispatch	M19 (O)	scientific, scoped FP32, reviewed	`49fa580a`
I27	DiffCSE	fixed-padding RTD dispatch	M19 (O)	scientific, scoped FP32, reviewed	`33b29a38`
I28	SimCSE++	baseline MLM dispatch	M19 (O)	scientific, scoped FP64, reviewed	`51d2db55`
I29	DiffCSE	RTD padding as negatives	M20 (L)	scientific, scoped FP64, reviewed	`33b29a38`

ID	Codebase	Violation	Group	Validation	Version
I30	SimCSE++	DCL term omitted[s]	M21 (O)	scientific, scoped FP64, reviewed	`51d2db55`
I31	Mamba	TP normalization group width	M22 (O)	scientific, scoped FP32, reviewed	† `e9594ce1`
I32	solo-learn	in-loss batch norm	M23 (L)	scientific, scoped FP64	`9187ea39`
I33	Mask2Former	weighted-CE local mean	M24 (L)	formal, scoped	‡ `9b0651c6`
I34	Barlow Twins	correlation all-reduce backward	M25 (R)	formal, scoped	‡ `8e8d284c`
I35	VICReg	tail-batch covariance denominator[s]	M26 (L)	formal, scoped	`4e12602f`
I36	OneFormer	weighted-CE per-image weights	M24 (L)	formal, scoped	‡ `4962ef6a`
I37	SASSHA	pre-training SUM reduction	M27 (R)	case-level, reviewed contract	`9c2c5611`
I38	SimCSE	DDP auxiliary weight	M28 (R)	scientific, scoped FP32/FP64	‡ `49fa580a`
I39	DiffCSE	DDP auxiliary weight	M28 (R)	scientific, scoped FP32/FP64	‡ `33b29a38`

Prospective additions. The following four partition-specific records have scoped dual-host repair closure (cf. Appendix F.6). Their identifiers preserve the original inventory; no new mechanism-group assignments are made.

R4-FE, FlagEmbedding: relative KL-distillation loss scale under data parallelism; audited commit `fd1a2bdf`.

R4-GR, GritLM: relative generation/representation loss scale; audited commit `97106810`.

R4-SL, SLIP: missing CLIP key-side backward contributions; audited commit `c6faf5d0`.

R4-DP, Diffusion Policy: accumulation-dependent update boundary and EMA clock; audited commit `5ba07ac6`.

Contract sources by record. The links below identify the source definitions at the audited revisions. Where those definitions leave the statistical scope or update boundary open, the text states the audit's convention. These choices and their support domains are part of the contract; a repair does not by itself establish the original authors' intent. Full revisions and evidence-file hashes are recorded in `data/audit_source_bindings.json`.

Records	Source and audited contract
I01	Qwen3-MoE configuration defines the auxiliary coefficient. The audit holds the composite CE/router objective fixed and restricts the partition pair to mirrored routing distributions.
I02	Megatron router supplies the approximate-prefix computation. The audit uses cumulative valid-token mass within that approximation and a fixture whose reference is partition invariant; it does not require exact-global routing statistics.
I03	OpenCLIP README defines accumulation as a larger effective batch. For the experimental NaFlex path, the audit explicitly uses one window-mass coefficient with linear scaling; the README alone does not specify this unequal-mass case.
I04	AcceleratedScheduler docstring ties scheduler progress to actual optimizer steps, including overflow skips.
I05	DeepSpeed clipping API defines the norm of the parameter gradients. The audit applies the standard L2 norm to the unique logical vector, counting replicated dense coordinates once and each expert on its owner.
I06, I14	DINO and DenseCL specify the EMA, center, and queue recurrences. The audit fixes their clock at one commit per logical optimizer update, reading the old auxiliary state throughout the window.
I07	KL-Cov loss definition specifies covariance-based token selection and its quota. The audit fixes both over the logical minibatch before slicing the selection mask into microbatches.
I08, I16	bitsandbytes and TorchAO specify size-based low-bit state policies. The audit binds eligibility and block identity to the original logical parameter; flattening or sharding must preserve that policy.
I09, I18	NeMo distillation losses define the CE/KD ratio and teacher scoring. The audit evaluates the detached ratio or teacher choice over the entire logical window; qualification covers the train body and teacher IPC in the stated singleton profile.

Records	Source and audited contract (continued)
I10, I11	Canzona README describes full-matrix optimizer computation under uniform sharding. The reviewed contract preserves each parameter's complete optimizer history when the execution owner changes.
I12	Adam-mini README assigns one adaptive rate to each parameter block. The audited default scalar block retains its whole-block second moment under FSDP sharding.
I13	Q-GaLore training entry configures trainable floating-point parameters alongside quantized ones. The audited optimizer step includes every configured parameter with a gradient.
I15	veRL GMPO loss takes a sequence mean. The audit fixes its denominator to the logical minibatch sequence count, including when dynamic microbatches have unequal sizes.
I17	BCO documentation and RunningMoments usage define the running reward baseline. The audit reads one pre-update baseline throughout the window and commits its moments once at the update boundary.
I19	Megatron vocabulary-parallel CE supplies the label-smoothing convention. The reference keeps that convention, including exclusion of the target class, and evaluates it over the full vocabulary.
I20	AReaL M2PO functional supplies the pinned full-batch behavior. The reviewed reference preserves it by global selection and reconciliation with the engine's original-mass weighting; this is not a claim about the authors' full-paper objective.
I21, I22	GaLore and Q-GaLore configure gradient accumulation and update counts. The audit accumulates a complete logical gradient before one optimizer/scheduler transition; I22 uses the frozen continuation profile with the common floating-hook repair.
I23, I24	APOLLO scheduler configuration supplies the declared warm-up and horizon. Both audited per-layer entries use their admitted GA1 setting and one scheduler tick per update.
I25	TRL GMPOTrainer selects the GMPO objective. Enabling Liger changes the execution path while the selected objective remains GMPO.
I26, I38	SimCSE README and composite loss specify the optional MLM objective and its coefficient. The audited contract preserves an enabled MLM term under padding changes (I26) and its relative weight under DDP (I38).
I27, I29, I39	DiffCSE README and composite loss specify RTD and its weight. The audit preserves enabled RTD under padding dispatch (I27), treats padding as outside the valid-token RTD mean (I29), and fixes its relative DDP weight (I39).
I28, I30	SimCSE++ README identifies the main recipe. I28 separately audits the optional MLM on the baseline entry; I30 preserves the configured DCL term on the main `train_re_vicreg.py` entry. These do not establish exposure in published runs.
I31	Mamba-2 constructor declares the grouping configuration. The audit preserves the unsharded normalization groups under contiguous, divisible TP sharding on the unfused path.
I32	solo-learn Barlow loss is audited against global-batch normalization, consistent with the official SyncBN configuration. Loss-local BN is isolated from the separate correlation-backward issue.
I33, I36	Mask2Former and OneFormer classification criteria supply the class weights. The audit explicitly chooses a global weighted-mean CE, retaining mask and Dice terms; it does not infer that this is the only possible author-intended objective.
I34	Barlow Twins loss and LARS define a global cross-correlation loss and the optimizer. The reference differentiates that loss and applies the same configured LARS update, including its bias/norm exclusions.
I35	VICReg covariance term is interpreted as sample covariance: its denominator is the actual sample count minus one, including a partial final batch.
I37	SASSHA pre-training recipe is audited under the reviewed mean-gradient contract with equal-mass ranks and unchanged optimizer settings. The SUM finding is conditional on this explicit contract, not an author acknowledgment.

D.3 Declared and executed computations

Table D.3 describes the computation changed in each of the 39 audit records: coefficients, statistics, objectives, or state-update rules. The verification column distinguishes native measurements

Table D.1: The audit by domain: 43 violations in 31 frameworks and research codebases, one entry per violation's mechanism. [s]Shared by every partition tested; [†]publicly reported before our audit; [‡]related public discussion. Per-record details: Appendix D.

Domain	Records	Codebase: mechanism of each violation
LLM training	8	Transformers: Qwen3-MoE auxiliary-loss scale. Megatron-Core: router token mass; TP label smoothing[†]. DeepSpeed: AutoEP clipping norm. Accelerate: fused-AMP skipped steps. NeMo RL and Automodel: CE–KD weights; multi-teacher selection. Mamba: normalization groups[†].
RL post-training	5	veRL: KL-Cov selection; GMPO sequence means[†]. TRL: BCO running moments; GMPO with Liger[s†]. AReaL: M2PO selection[†].
Optimizers	11	bitsandbytes: 8-bit state eligibility under FSDP1. TorchAO: the same under FSDP2. Adam-mini: block statistics[‡]. Canzona: SOAP owner state; Muon owner state. GaLore: per-layer update clock[†]. Q-GaLore: continuation clock[†]; floating-point parameter hook[s]. APOLLO: scheduler, in two entry points[s]. SASSHA: SUM reduction.
Self-supervised vision	5	LightlyTrain: DINO teacher and center; DenseCL queue. solo-learn: in-loss batch norm. Barlow Twins: correlation all-reduce[‡]. VICReg: tail-batch denominator[s].
Sentence embeddings	9	SimCSE: fixed-padding dispatch; DDP auxiliary weight[‡]. DiffCSE: fixed-padding dispatch; DDP auxiliary weight[‡]; RTD padding. SimCSE++: baseline dispatch; DCL term[s]. FlagEmbedding: cross-device KL distillation weight. GritLM: relative generation/representation loss scale.
Segmentation	2	Mask2Former: weighted-CE local mean[‡]. OneFormer: the same[‡].
Vision–language	2	OpenCLIP: NaFlex replay weights. SLIP: CLIP key-side backward omission.
Robot learning	1	Diffusion Policy: update boundary / EMA clock.
Total	43	37 partition-specific and 6 shared records

and repair checks from explicit executions of the modified computation. Shared mechanisms do not transfer experimental results between implementations. Record identities and audited versions follow Table D.2.

Here W is the data-parallel degree, p the tensor-parallel degree, E the expert-parallel degree, and k the microbatches per logical update. Repair checks compare affected and repaired implementations within the audited domain; full-training equivalence requires separate evidence. The table states the granularity of each comparison and reports inconclusive effects and failed equivalence tests.

Table D.3: Declared computation, executed change, and verification for all 39 implementation records. Related records share rows where useful; evidence remains implementation-specific. Maps apply under the stated conditions, not to arbitrary training configurations.

Record / system	Declared → executed computation	Verification and scope
I01 Transformers	Auxiliary coefficient λ → $W\lambda$ through world-size compensation.	Native gradient/repair checks. At $W = 4$, scores and final hashes match repaired 4λ in 15 archived pairs; post-hoc endpoints, mirrored rank distributions.
I02 Megatron-Core	Accumulated token mass → microbatches seen × current token count.	Native gradient gap $0.25 \to 2.3 \times 10^{-7}$ after repair. Approximate-prefix contract only; not exact-global statistics.
I03 OpenCLIP	Window-weighted replay gradients → microbatch scales; shared scale/bias gradients retained only in the last replay.	Native repair and softplus reference; two seeds, eight FP64 steps. Audited NaFlex path; no universal scalar map across parameters.

Continued on next page

Table D.3 continued: declared and executed computations

Record / system	Declared → executed computation	Verification and scope
I04 Accelerate	Scheduler counts successful updates → counts attempts, including fused-AMP overflow skips.	Explicit attempt-clock control matches recorded GPT2 losses, LR, RNG, updates, and final hashes; RTE also tested. Small quality effect; no GPU-count-dependence claim.
I05 DeepSpeed	Global norm $\sqrt{D + \sum_j q_j}$ → $E^{-1} \sum_j \sqrt{D + q_j}$. D: replicated dense mass; q_j: unique expert mass.	Clipping/update discrepancies $\approx 0.20 \to 0$ after repair. Two ranks, FP32, L2, ZeRO-0, active clipping.
I06 LightlyTrain DINO	One teacher/center commit per update → one per microbatch; teacher retention m^k if student is fixed.	Native state/gradient checks and boundary-commit repair. Center depends on microbatch statistics and order; no joint long-run map.
I07 veRL KL-Cov	Minibatch statistics and token quota → covariance/top-k recomputed per microbatch, selecting at least one each.	One versus eight microbatches selects 1 versus 8 tokens; global-mask repair closes the update gap. Native FSDP single-update scope.
I08, I16 bitsandbytes; TorchAO	Original-parameter precision eligibility → flattened/local-shard size selects the 8-bit-state branch.	Separate native state/update repairs: I08 FSDP1, I16 FSDP2; I08 also has original-identity reference and FP32 control. Policy-specific thresholds.
I09 NeMo RL + Automodel	Window CE/KD ratio α → microbatch ratios α_j: $\alpha \sum_j \nabla K_j \to \sum_j \alpha_j \nabla K_j$.	Train-body/teacher-IPC gradient gap $0.0166 \to 6.6 \times 10^{-7}$ after repair. Excludes full driver, actual FSDP, and Ray construction.
I10, I11 Canzona SOAP; Muon	Optimizer history follows parameter owner → new owner reads stale/empty local history; SOAP may reinitialize and skip.	Warm-state parameter gaps 0.0986 (SOAP), 0.0109 (Muon), both repaired to zero. Two ranks, uniform tensor shards.
I12 Adam-mini	Whole-block second moment → fragment-local statistics when device visibility suppresses reduction.	Explicit fragment/whole controls, nonzero-state continuations, and FP64 formula checks. Audited default FSDP blocks; five-seed quality effect inconclusive.
I13 Q-GaLore	Update all parameters with gradients → unregistered hooks freeze floating-point parameters, $\Delta\theta = 0$.	Seven parameters have gradients but no steps; hook repair restores all 21 updates and reference state. Shared error; full quality repair also changes the clock.
I14 LightlyTrain DenseCL	Read old queue; commit EMA/queue once → commit per microbatch, changing later reads.	GA1/4 checks: repaired state exact, gradient residual 1.5×10^{-7}. Joint EMA/queue rule; momentum alone is insufficient.
I15 veRL GMPO	Logical-minibatch sequence mean → sum of local means; unequal microbatches also reweight sequences.	Native 1×8 versus 8×1 comparison and global-count repair. Audited GMPO sequence-mean path.
I17 TRL BCO	Baseline moments committed once → later microbatches use moments updated by earlier ones.	Native gradient/update checks; boundary repair gives zero update residual. Joint baseline/objective change; no long-run mapping.
I18 NeMo RL + Automodel	Teacher selected from window scores → selection per microbatch despite cross-rank score reduction.	Native train-body check and repair for teacher selection; does not cover all continuous weighting modes.
I19 Megatron-Core	Global-vocabulary label smoothing → shard-local means/normalizers alter the smoothing term.	Native component and repair checks with nonzero smoothing. No general $\alpha \mapsto \alpha'$ map asserted.
I20 AReaL M2PO	Window trust set and original token mass → microbatch reselection; pre-selection mass multiplies a post-selection mean.	Mask counterexample, native check, global-mask/original-mass repair. Fixed implementation contract; no full RL-task result.

Continued on next page

Table D.3 continued: declared and executed computations

Record / system	Declared → executed computation	Verification and scope
I21, I22 GaLore; Q-GaLore	Accumulate k, clip and update once → k transitions; hooks consume gradients before outer clipping.	I21: 128 steps in four 7B updates, quality/resource tests. I22: cached-projection continuation and quality tests after common hook repair. Loss/k is not one Adam update.
I23, I24 APOLLO	Declared scheduler warm-up/horizon → both doubled, with one clock tick per update.	Native LR/full-state repairs in both per-layer entries. Three-seed quality experiment belongs to I23 only.
I25 TRL	Selected GMPO objective → Liger dispatch executes GRPO.	Native objective comparison and repair. Audited integration; shared across tested partitions.
I26–I28 SimCSE; DiffCSE; SimCSE++	Enabled MLM/RTD objective → fixed-padding collator omits auxiliary inputs; objective not computed.	I26: three-coefficient check and full-training on/off saved-state and score equality. I27: small-model explicit RTD-off single-step match. I28: baseline-script dispatch/repair check.
I29 DiffCSE	RTD mean over valid tokens → padding labeled negative and included in the mean.	Native loss/gradient repair; two-seed best-dev/endpoint readouts. Changes loss content and denominator, not just scale.
I30 SimCSE++	Configured DCL term → zero contribution despite nonzero coefficient.	Scoped native objective/repair checks on the main entry. Historical published-run exposure not established.
I31 Mamba-2	Normalization in G groups → pG groups in unfused TP; B/C grouping unchanged.	Native $p = 1, 2, 4$ checks and explicit eight-group forward/gradient tests. New five-seed TP4/16-group mimic test passes the preset ± 0.005-nat endpoint margin; historical TP2 had no preset margin. Contiguous divisible shards; not trajectory-exact.
I32 solo-learn	Global-batch loss normalization → temporary loss-local BN uses rank-local statistics.	FP64 factor isolation: BN repair restores partition agreement; joint BN/reduction repair matches reference. Distinct from I34.
I33, I36 Mask2Former; OneFormer	Global weighted-mean classification → image weight $M/(WM_r)$; M, M_r: global/local normalization masses. Mask/Dice unchanged.	I33: gradient prediction and explicit replay, saved states/PQ equal through 10k steps. I36: separate scoped qualification. Per-image domain; no transfer of training results.
I34 Barlow Twins	Declared LARS on global-loss gradient → gradient $1/W$: adapted decay $W\mu$, excluded bias/norm LR η/W.	Native group checks; mapped warm-up probe scores match recorded precision. Reachable-state residuals nonzero but below threshold. Audited LARS, matched states; not trajectory-exact.
I35 VICReg	Covariance denominator $n-1$ → $B-1$; squared penalty factor $[(n-1)/(B-1)]^2$.	Measured 0.04 at $n = 4, B = 16$; count repair, covariance reference, full-batch control. Tail batches; factor is not total-gradient/quality scaling.
I37 SASSHA	Mean gradient across equal-mass ranks → SUM; conditional update factor $\approx W^{1-\alpha}$.	CPU formula and two-host native SAM checks. Approximation requires no clipping, negligible ϵ, cold or suitably scaled state.
I38, I39 SimCSE; DiffCSE	Relative auxiliary weight λ → $W\lambda$, from $g_C/W + \lambda g_A$. Run map needs common-scale/optimizer conditions.	Both: native checks with matched views/auxiliary mass. I38: five-seed weight effect, failed preset mapping-equivalence test. I39: common adaptations; no inherited task result.

D.4 COUNTING UNITS

The paper uses several counting units, and they do not add up (Table D.4). An *implementation record* is one method or entry point closed by a single-update repair; seeds, precisions, repeats on the same

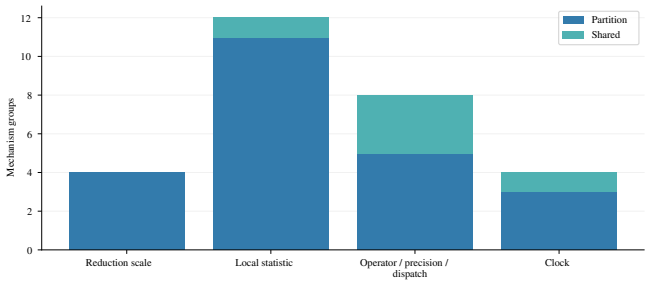

Figure D.1: The original cohort's 28 mechanism groups by class and type. Groups include publicly known mechanisms; the figure estimates neither prevalence nor novelty.

host, second hosts, and later training runs add no records. A *mechanism group* collects records that share responsible code, contract, and repair; groups include publicly known mechanisms, and no global adjudication of merges or splits across groups was made, so the 28 groups are not 28 new independent causes. A *primary target* is an audited framework or project, not a count of GitHub repositories: NeMo RL with Automodel is one target, and dependencies such as PyTorch or Liger are not targets.

Table D.4: Counting units. Mechanism and scope groups describe only the original 39 records.

Unit	Count	Meaning
Implementation records	43	37 partition-specific and 6 shared
Mechanism groups	28	original 39 records: 23 partition-specific and 5 shared; includes known mechanisms
Primary targets	31	audited frameworks and research codebases
Scope groups	35	original 39 records grouped by implementation scope
Formal, scoped, or case-level records	43	heterogeneous levels (cf. Appendix A.4; cf. Appendix F.6), not uniform certification
Original-cohort dual-host follow-ups	3	Mask2Former, official Barlow Twins, official VICReg; included above

D.5 Prior public reports and search scope

Table D.6 preserves the earlier mechanism-group snapshot and lists the public sources behind its ten marked groups. A dagger means the group's mechanism was publicly reported before our audit; a double dagger means related public discussion that does not report the audited instance. The markers come from case-by-case reviews of each project's public issues and pull requests; we did not run an exhaustive search. The remaining 18 groups therefore carry no marker and no claim of first discovery. Only reports by others are listed here; our own disclosures are summarized separately (cf. Appendix D.7).

Subsequent review by implementation record. A bounded follow-up search, recorded on 24 September 2026, classified the original 39 records as follows: for 26, no earlier report of the same specific error was found within the search scope; 7 had related prior public evidence; and 6 had direct public precedent or were already classified as publicly known. These are record counts, not mechanism-group counts. The search covered visible issues, pull requests, and issue comments in selected official repositories, supplemented by targeted searches and earlier archives; it was not exhaustive. No verified pre-search freeze is available for this follow-up. Absence of a located report does not establish first discovery, and the four prospective additions are outside this review.

D.6 Defects outside the partition contract

Two confirmed defects are real but fall outside the partition contract, so they are not among the 39 records. In LoRA-GA with AdaLoRA (PEFT 0.10), the rank-allocation call is never made: training

Table D.5: Where a partition leaks in and what it rewrites. Counts cover the original 39 records. Four prospective additions are reported separately; mechanism-group deduplication has not been extended to these additions. Examples are representative. Partition groups are classified by a broken premise; shared groups by the mechanism they resemble. A mechanism may break several premises. Full cases are documented separately (cf. Appendix D). †Publicly known before our audit; ‡related public discussion. Unmarked groups carry no first-discovery claim.

Mechanism (premise broken)	*Groups: partition / shared*
Reduction scale (exact merge) *What is rewritten:* objective coefficients or optimizer hyperparameters (Prop. 2). SimCSE/DiffCSE DDP‡ ($\lambda \mapsto W\lambda$); Barlow Twins‡ (biases and normalization: $\eta \mapsto \eta/W$; adapted parameters: $\mu \mapsto W\mu$)	**4 / 0**
Local statistic (exact merge, global information) *What is rewritten:* per-example or per-token weights, denominators, selections, norms. DeepSpeed AutoEP clipping norm; veRL KL-Cov (top-k quota $1 \mapsto 8$); Mask2Former/OneFormer‡ ($\omega_r = M/(WM_r)$); *shared:* VICReg tail-batch denominator	**11 / 1**
Partition-dependent operator, precision, or dispatch (partition-free evaluation) *What is rewritten:* architecture, optimizer-state precision, statistic blocks, owner state, which objective is computed. Mamba-2† ($G \mapsto pG$); bitsandbytes/TorchAO 8-bit state eligibility; SimCSE-family fixed padding (auxiliary loss omitted); *shared:* Q-GaLore floating-point parameter hook	**5 / 3**
Clock (single transition) *What is rewritten:* number and timing of optimizer and auxiliary-state transitions. GaLore/Q-GaLore† per-layer steps ($1 \mapsto k$ per update, clip bypassed); Accelerate fused AMP (scheduler advances on skipped steps); *shared:* APOLLO scheduler	**3 / 1**
Original audit: 23 partition / 5 shared mechanism groups; 39 records in 27 codebases.	

Table D.6: Public sources for the marked mechanism groups in the earlier snapshot.

Group	Mark	Public source	Scope of the mark
veRL GMPO sequence means	†	veRL PR 5614	same code mechanism
Megatron-Core TP label smoothing	†	Megatron-LM issue 737, PR 5522	same mechanism; earlier reports, not our disclosure
AReaL M2PO selection	†	AReaL PR 1546	known composite mechanism within the recorded scope
GaLore/Q-GaLore update clock	†	GaLore issue 62	accumulation and scheduler units; not every Q-GaLore variant
TRL GMPO with Liger	†	TRL issue 6808	same dispatch mechanism
Mamba-2 TP group width	†	state-spaces/mamba PR 908	same mechanism; the PR need not have merged
Adam-mini block statistics	‡	Adam-mini issues 7, 14, 35	related device-visibility and reduction discussion
Mask2Former/OneFormer weighted CE	‡	Mask2Former issue 203	related classification-weight discussion
Barlow Twins correlation all-reduce	‡	barlowtwins issue 8	related discussion of the same collective
SimCSE/DiffCSE DDP weight	‡	SimCSE issue 212	related gather and normalization discussion

keeps 24 active singular components where the target is 8, and a repair that restores the allocation schedule changes the update by a relative distance of about 0.49. Two partitions of the same update agree to within 10^{-5} before and after the repair, so this is a fault every partition shares, visible only against the reference. The second is GaLore's outer gradient clip, which cannot see gradients that per-layer hooks have already consumed: a shared observation with isolated evidence, recorded apart from the clock mechanism it accompanies.

D.7 Disclosure

Section 5 summarizes the disclosure snapshot of 18 September 2026. By then, the LightlyTrain maintainers had acknowledged the DINO problem and chosen to fix it first in the underlying LightlySSL library, TRL had closed the BCO issue when it removed the BCO module, and an independent contributor had reproduced the DeepSpeed AutoEP fault. Since then, the Megatron-Core maintainers have acknowledged the label-smoothing problem and begun a fix. None of the proposed patches had been merged at the snapshot. Here we add how the snapshot maps onto the audit. The 26 reported cases are covered by 26 issues and 24 patches, because the two NeMo cases and the two Q-GaLore cases each share a patch. Their 19 repositories correspond one to one to 19 of the 27 targets, with Megatron-LM standing for Megatron-Core, NeMo RL for NeMo RL with Automodel, and FSDP-Canzona for Canzona. The eight targets not yet reported at the snapshot are SimCSE, Mamba, solo-learn, Mask2Former, OneFormer, the official Barlow Twins and VICReg, and SASSHA. Reports cover cases, not every record: some records of a reported target, such as DiffCSE's dispatch and data-parallel records, were not yet part of a report. Upstream status was not refreshed after the snapshot except where §5 says so, and local repair closure (cf. Appendix A.4) is independent of upstream acceptance.

In the later snapshot of 25 September 2026, 26 deduplicated issues covered 28 of the original 39 implementation records. This record-level coverage is separate from the 18 September codebase-level snapshot above.

E Checker Evaluation

Table E.1: PartitionCheck on the original 28-row roster and a separate archival replay. The roster is constructed, not sampled, so its rates describe these rows only.

Measure	Result	Composition and scope
Faults detected	14/14	10 injected, 4 real
Rows assigned their correct class	28/28	10 partition-specific, 4 shared, 11 clean, 2 boundary, 1 non-comparable
Shared faults, reference axis only	4/4	injected
First divergent stage, complete trace	12/14	2 real adapters record no earlier stage
Correct abstention before execution	3/3	1 non-comparable plan, 2 boundaries
False alarms on benign runs	0/9	8 held out; one-sided 95% Wilson bound 0.23
Executed rows that fail to repeat	0/25	same-plan repeats judged first
Adapter size	236 lines	median non-blank lines, 3 release adapters
Added step time, CI / full trace	0.14% / 0.94%	medians, 3 adapters; full-trace maximum 40.4%
Archival faults detected (E10b)	36/36	separate 108-row evidence-replay roster
Matched benign alarms (E10b)	0/36	one-sided 95% Wilson upper bound 6.99%
Profile-drift abstention (E10b)	36/36	constructed unsupported probes

E.1 The frozen roster

The evaluation roster has 28 rows, fixed before evaluation (Table E.2): ten injected faults on the tiny exact adapter, four real faults (the three release lanes MG01, HQ03, and C09-R3, and a historical Transformers fault, TH02), nine benign test cases, two repaired or fixed arms of the historical case, and three plans that must be rejected before execution. Eight of the nine benign test cases are held out from calibration; the ninth, the historical negative control, is a fresh-process rerun of calibration fixture B001, and the frozen label that counted it as held out is corrected here. The roster and the 39 audit records are different sets, and the roster is constructed rather than sampled.

Table E.2: The 28-row roster. PC: partition consistency; RC: reference conformance. "Trace" marks whether every stage before the first divergence was recorded. *A fresh-process rerun of a calibration fixture, hence not held out.

Row	Source	Admission	PC	RC	Class	First divergent stage	Trace	Held out
Tiny fault F01	injected fault	executed	violating	violating	partition specific	objective coefficients	yes	no
Tiny fault F02	injected fault	executed	violating	violating	partition specific	objective mass	yes	no
Tiny fault F03	injected fault	executed	violating	violating	partition specific	local gradient	yes	no
Tiny fault F04	injected fault	executed	violating	violating	partition specific	unscaled gradient	yes	no
Tiny fault F05	injected fault	executed	violating	violating	partition specific	reduced gradient	yes	no
Tiny fault F06	injected fault	executed	violating	violating	partition specific	clip coefficient	yes	no
Tiny fault F07	injected fault	executed	conforming	violating	shared	local gradient	yes	no
Tiny fault F08	injected fault	executed	conforming	violating	shared	step counter	yes	no
Tiny fault F09	injected fault	executed	conforming	violating	shared	parameter delta	yes	no
Tiny fault F10	injected fault	executed	conforming	violating	shared	parameter delta	yes	no
Accelerate fused AMP (C09-R3)	real fault	executed	violating	conforming	partition specific	overflow skip flag	yes	no
Transformers Qwen3-MoE (HQ03)	real fault	executed	violating	conforming	partition specific	objective component	no	no
Megatron-Core token mass (MG01)	real fault	executed	violating	conforming	partition specific	routing frequency, step 2	no	no
Hist. Transformers, affected	real fault	executed	violating	conforming	partition specific	objective coefficients	yes	no
Tiny benign B004	no fault	executed	conforming	conforming	none	–	yes	yes
Tiny benign B008	no fault	executed	conforming	conforming	none	–	yes	yes
Tiny benign B012	no fault	executed	conforming	conforming	none	–	yes	yes
Tiny benign B016	no fault	executed	conforming	conforming	none	–	yes	yes
Tiny benign B020	no fault	executed	conforming	conforming	none	–	yes	yes
Tiny benign B024	no fault	executed	conforming	conforming	none	–	yes	yes
Tiny benign B028	no fault	executed	conforming	conforming	none	–	yes	yes
Tiny benign B032	no fault	executed	conforming	conforming	none	–	yes	yes
Tiny plan F11 (drops an atom)	no fault	rejected	–	–	non-comparable	–	–	no
Tiny plan F12 (no global clip statistic)	no fault	rejected	–	–	boundary	–	–	no
Hist. Transformers, repaired	no fault	executed	conforming	conforming	none	–	yes	no
Hist. Transformers, fixed upstream	no fault	executed	conforming	conforming	none	–	yes	no
Hist. Transformers, negative control*	no fault	executed	conforming	conforming	none	–	yes	no
Boundary control (TT03)	no fault	rejected	–	–	boundary	–	–	no

E.2 INJECTED FAULT FAMILIES

The injected faults are trace-level transforms applied after execution to an immutable benign capture of the tiny adapter: a deterministic scale or increment to one or more stages of one arm (a partition-specific fault) or of both arms (a shared fault); the untouched capture is the repair arm (Table E.3). The injector reads only trace structures and shares no oracle code, and the scorer receives observations with the fault class, expected stage, and expected verdicts removed. Every injected fault changes its stage by at least 25%, so the roster says nothing about sensitivity to violations near the threshold.

E.3 CALIBRATION

Every tolerance belongs to one profile and is fixed before the formal comparison, from controls with a known outcome; the formal fixture and all fault labels stay out of calibration. The profiles fall into two regimes.

Small-model profiles. On the tiny profile, the per-control statistic is the maximum stage distance over the partition pair and the stage family, and the threshold is the largest of 24 fitted controls; every fourth of 32 benign controls (8) is held out, and the held-out controls use a different token

Table E.3: The twelve frozen fault families. Arms: B only (partition-specific) or A and B (shared).

Family	Mechanism	Transform	Expected stage	PC / RC
F01	partition-local mean of microbatch means (B)	$\times 0.75$	objective coefficients	violating / violating
F02	objective mass counted twice (B)	$\times 0.5$, $\times 2$	objective mass	violating / violating
F03	partition-local gradient scale (B)	$\times 2$	local gradient	violating / violating
F04	loss unscale omitted (B)	$\times 8$	unscaled gradient	violating / violating
F05	reduced gradient divided twice (B)	$\times 0.5$	reduced gradient	violating / violating
F06	clip coefficient computed per partition (B)	$\times 0.5$	clip coefficient	violating / violating
F07	undeclared gradient scale (A and B)	$\times 2$	local gradient	conforming / violating
F08	extra optimizer transition (A and B)	$+1$, $\times 2$	step counter	conforming / violating
F09	parameter delta applied twice (A and B)	$\times 2$	parameter delta	conforming / violating
F10	weight decay in the wrong transition (A and B)	$\times(-1)$	parameter delta	conforming / violating
F11	plan drops one logical atom	–	rejected before execution	non-comparable
F12	global clip statistic unavailable	–	rejected before execution	boundary

stride (7) from the fitted ones (1, 3, and 5). The fitted cross-partition threshold is 2.6×10^{-14}, with no held-out alarm. The historical Transformers profile fits 24 controls the same way (threshold 1.8×10^{-4}) with no held-out alarm. The split and the control generator are deterministic, so the held-out controls form a constructed test suite rather than a sample of future test executions.

Framework lanes. On the three release lanes, each threshold is the larger of a fixed floor and ten times the largest control distance over three held-out fixtures, with the formal fixture excluded (Table E.4). The controls differ by lane. HQ03 uses two benign modes, the auxiliary term switched off and the component-aware repair, on three fixtures: six cross-partition comparisons. MG01 runs the repaired code and the oracle under both partitions on three fixtures, twice each: 24 processes. C09-R3 uses fused, unfused, and reference controls on three fixtures; its primary checks, the skipped-step flag and the scheduler clock, are exact discrete comparisons, and the numeric thresholds serve as guards. No lane has an independent held-out benign test beyond these fixtures.

Table E.4: Frozen thresholds on the framework lanes: relative L_2 threshold and the largest control distance it was fitted to. Where a floor dominates, the threshold exceeds ten times the control maximum.

Lane	Axis	Control maximum	Threshold
HQ03	router gradient (primary)	3.5×10^{-7}	3.5×10^{-6}
HQ03	non-router gradient	2.0×10^{-7}	2.0×10^{-6}
HQ03	parameter delta	1.0×10^{-6}	1.0×10^{-5}
MG01	gradient	1.7×10^{-6}	1.7×10^{-5}
MG01	objective	6.6×10^{-8}	10^{-5} (floor)
MG01	parameter delta	3.1×10^{-5}	10^{-3} (floor)
MG01	routing frequency, step 2	0	10^{-6} (floor)
C09-R3	parameter delta (guard)	1.3×10^{-6}	1.3×10^{-5}
C09-R3	objective and pre-trigger gradient (guards)	0	10^{-6} (floor)

What the envelopes support. If calibration and test statistics were jointly exchangeable, a benign run would exceed the maximum of N calibration statistics with probability at most $1/(N+1)$; scaling the maximum by a factor of at least one or adding a floor cannot raise that probability. The bound is marginal, averaged over calibration and test draws, not conditional on one realized calibration set, and constructed controls do not establish exchangeability; with $N = 3$ it would be 25%, which is neither a measured false-alarm rate nor an established bound. The envelopes are therefore frozen empirical tolerances. Their evidential weight lies in the margins: the normalized detection statistic of the reported violation is about 42,000 times the threshold on HQ03 and between 250 and 500,000

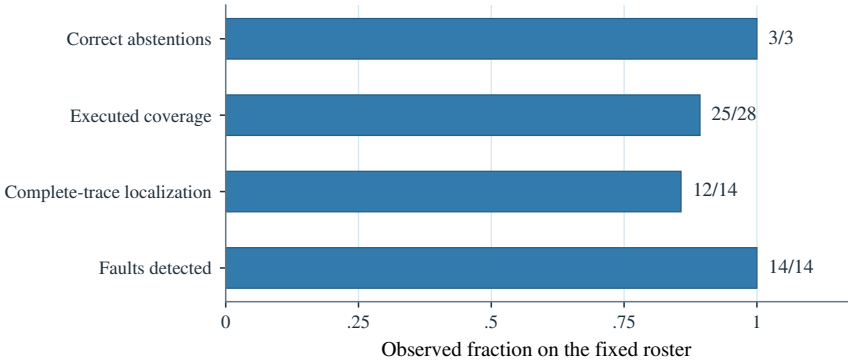

Figure E.1: Counts on the fixed roster, each with its own denominator: correct abstentions (3/3), executed coverage (25/28), localization with complete earlier trace (12/14), and detection (14/14). Benign false alarms were 0/9, of which 8 are held out. No general recall or false-alarm rate is implied.

times it across MG01's axes; C09-R3's violation is a discrete flag. Same-plan repeats have statistic 0 on all three lanes, and the reference statistics of the conforming arms stay below threshold (0.096 on HQ03, 6.3×10^{-11} on MG01). No sweep of small, near-threshold violations was run.

E.4 Multiplicity

Multiplicity is handled by construction. Because the per-control statistic is a maximum over the tested pair and the stage family, one threshold governs the whole family on the small-model profiles. Each framework lane predeclares a primary axis and treats the others as diagnostics: the router gradient for HQ03; the skipped-step flag for C09-R3, with the scheduler clock as confirmation; and four signal axes for MG01, where a violation requires both the absolute and the relative threshold. No correction is applied across roster rows. The only roster-level inference is a one-sided 95% Wilson upper bound: 0.23 for false alarms in the nine benign test cases, and 0.098 for nonrepeatability over the 25 executed rows. These bounds describe the roster, and the acceptance rule uses point estimates.

E.5 Detection, classification, and localization

All 14 faults are detected and all 28 rows receive their correct class: 10 partition-specific, 4 shared, 11 without violation, 2 boundaries, and 1 non-comparable (Figure E.1). The four shared injected faults pass the partition comparison, their cross-arm distances staying inside the 2.6×10^{-14} envelope, and only the reference oracle reports them. The first divergent stage matches the expected stage in all 14 faults, but a stage counts as localized only when every earlier stage was recorded, which gives 12 of 14: the HQ03 adapter does not record local and reduced gradients separately, and the MG01 adapter does not split local from reduced gradients. In both, the alarm fired at the expected stage and was stable across blocks; what cannot be certified is that no earlier stage diverged. The three plans rejected before execution are rejected for the right reason. The two repaired or fixed arms and all nine benign test cases conform on both axes, and no executed row fails to repeat.

E.6 Overhead

Table E.5 reports the added step time of the three release adapters in CI mode and with full traces, each from five measurements per mode, converted from time ratios as $(\text{ratio} - 1) \times 100\%$; memory ratios were 1.0 throughout (Figure E.3). Negative estimates reflect measurement noise, not a speedup, and the maximum is kept. The adapters have a median of 236 non-blank lines.

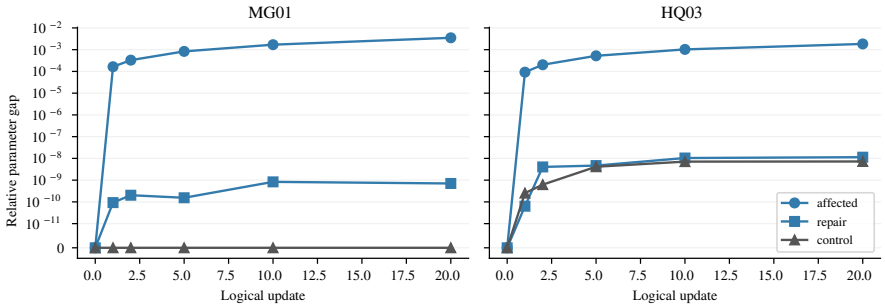

Figure E.2: Short trajectories of the MG01 and HQ03 release lanes: relative parameter gap of the affected, repaired, and control arms over 20 logical updates. Points are block averages of technical repeats with one scientific seed, not independent-seed uncertainty.

Table E.5: Added step time of the release adapters.

Lane	CI mode	Full trace	Trace size (bytes)
C09-R3	$+0.14\%$	$+0.94\%$	4,455
HQ03	-1.13%	-0.07%	4,499,738
MG01	$+10.53\%$	$+40.41\%$	783

E.7 Expanded archival evaluation and formal follow-up

The separate E10b roster contains 108 rows: 36 audited violations, 36 matched documented repair or negative-control configurations, and 36 constructed source/profile-drift probes. Twenty-five mechanism-specific adapter profiles replay archived evidence through the byte-frozen E10 scorer. All 36 faults are detected, all 36 benign configurations pass, and all 36 drift probes are rejected. Localization is correct at all 36 earliest adapter-visible logical boundaries, not at source-code lines. These results do not enlarge the original 28-row denominator or measure recall on unseen code. The one-sided 95% Wilson upper bound for 0/36 benign alarms is 6.99%; the frozen acceptance gate uses the point estimate, so it does not establish a population rate below 5%.

The same follow-up qualifies Mask2Former, official Barlow Twins, and official VICReg on two physical hosts, each with two RTX 5090 GPUs. Each case and host uses 24 fitted and eight held-out healthy pairs, followed by three formal blocks in AB/BA/AB order; local reanalysis of each returned archive reproduces the verdicts. All three pass within their small FP32, single-update domains. Mask2Former passes in a new confirmation after an earlier affected signal was observed; its earlier failed attempts remain archived. The cross-host comparator was amended after the second host's results were visible: residuals inside each host's frozen envelope count as the same null class; other comparisons retain their original tolerances. This outcome-aware amendment changes neither the per-host envelopes nor their verdicts, and is part of the reported qualification.

The 25 profiles require a median of 11 non-blank glue lines, plus 565 shared-engine lines counted once; this differs from the original release-adapter accounting. Across the three native follow-up cases, maximum added step time is 0.043% in CI mode and 5.71% with full traces, with a peak CUDA-memory ratio of 1.0. Seven warm-cache repetitions of the 36-record archival replay take median times of 0.099 s and 0.116 s respectively; these CPU replay times are separate from native training overhead.

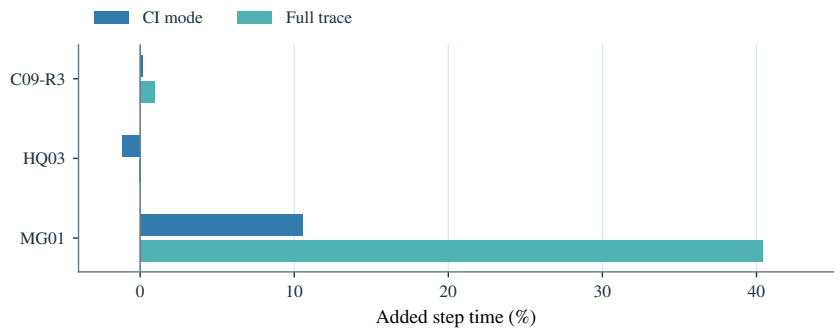

Figure E.3: Added step time per release adapter, CI mode and full trace (five measurements per mode).

F Full Results of Consequence Studies

This appendix details the consequence studies, with each study's role in the preregistration. Intervals are 95% unless marked; n counts seeds or seed pairs. Table F.1 summarizes the studies of §4.3, and Figure F.1 shows the mapped-run tests of §4.2 as paired differences in native units.

Table F.1: When rewrites change model quality—and when they do not. Confidence intervals are 95% unless marked; n counts seeds or seed pairs, not repeated evaluations.

Case	Contrast	Estimate [CI], n	Evaluation role
Mamba-2, 134M, TP4	Affected − repaired test NLL, evaluated as declared	0.189 $[0.170, 0.209]$ nats (97.5%); 5	Primary, passed (MPE 0.02)
Mamba-2, 134M	TP4 repaired − TP1 anchor test NLL	-0.0007 $[-0.0015, 0.0001]$ nats; 5	Primary, passed (within ± 0.005)
SimCSE, BERT-base	Two GPUs, λ=0.1 − one GPU, λ=0.1, loss $\times \frac{1}{2}$	STS -3.68 $[-4.56, -2.79]$; 5 new	New primary, passed
SimCSE, fixed padding	MLM restored − MLM off	Transfer $+0.46$ $[0.08, 0.83]$ (97.5%); STS -13.08 $[-14.70, -11.46]$; 5	Transfer primary, not passed (MPE 0.25); STS secondary
GaLore family, 60M LLaMA	Declared − original validation NLL	GaLore -0.21; Q-GaLore -4.12; APOLLO $+0.16$ nats; 3	Descriptive
Barlow Twins, ResNet-50, 2,048 steps	Repaired − raw backbone top-1	W=2: $+1.13, +0.90, +1.02$; W=4: $+1.74, +1.59, +1.45$ points; 3	Exploratory, within warm-up
Mask2Former, R50, 10k steps	Raw − repaired PQ	$+0.02, -0.03$ points; 2	Prospective 1-PQ gate, not reached

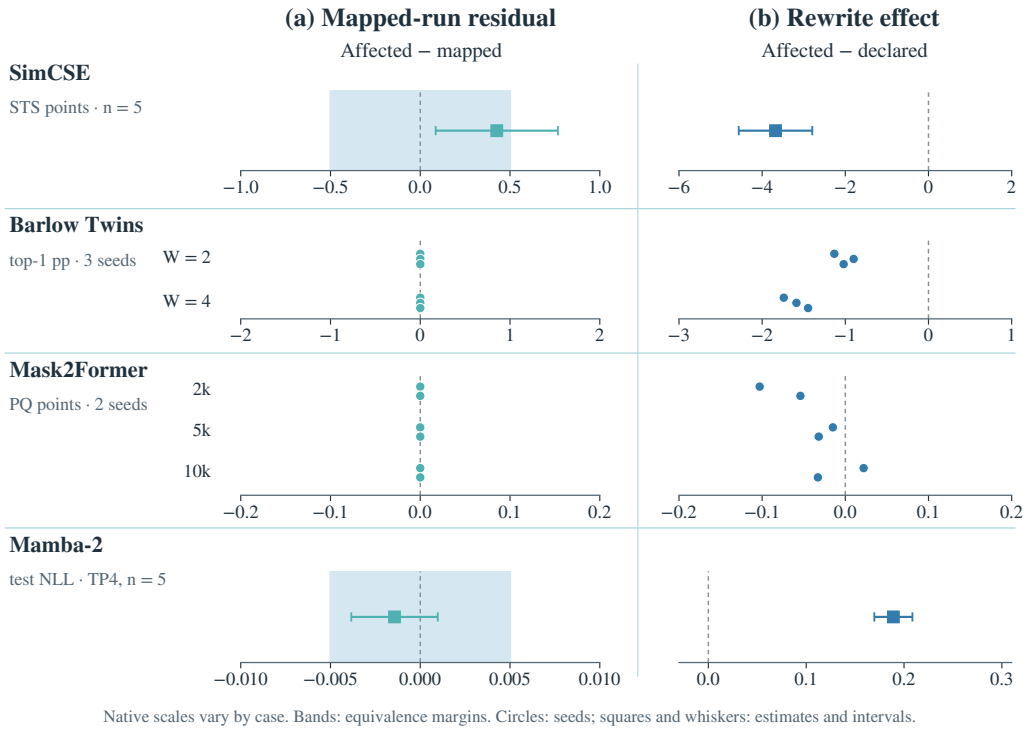

Figure F.1: Testing mapped runs and their consequences, in native units. Left: affected minus an explicit implementation of the rewrite (expected near zero). Right: affected minus the declared-value implementation. SimCSE: 5 new seeds, 90% CI left (the ± 0.5 equivalence test fails), 95% right (primary). Barlow Twins: 3 seed points at $W = 2, 4$ (exploratory, 2,048 steps). Mask2Former: 2 seeds, three saved checkpoints. Mamba-2: TP4, 5 seeds; native 16-group NLL and 95% CI left, declared four-group NLL and 97.5% CI right; both preregistered tests pass.

F.1 SimCSE

Layout study. BERT-base trained on 1M Wikipedia sentences for 15,625 steps at a global batch of 64, three seeds, with $\lambda = 0.1$. Three arms were fixed in the protocol before any run: the affected configuration (two GPUs, declared λ), the mapped configuration (one GPU, λ doubled to 0.2, loss scaled by one half), and the declared configuration (one GPU, declared λ, same loss scale). The written prediction was that the affected run matches the mapped arm, and the protocol explicitly makes no claim that it matches an unscaled $\lambda = 0.2$ run under arbitrary Adam ϵ, state, or clipping. Against the mapped arm, the affected run differs by -0.02 STS points ($[-0.91, 0.87]$); against the declared arm by -3.59 STS ($[-4.74, -2.44]$). Both are secondary contrasts.

Independent confirmation. A separately preregistered study repeats the same three configurations with five new seeds (631, 641, 647, 653, 659), the same corpus and 15,625-step budget, without pooling the earlier results. All 15 training endpoints and their frozen evaluations are complete. The sole primary contrast, affected minus declared-weight control, is -3.6789 STS points ($[-4.5635, -2.7943]$, $p = 0.000321$), passing the two-sided 0.05 test. Affected minus mapped is $+0.4262$ points (90% CI $[0.0859, 0.7664]$); the preregistered TOST at ± 0.5 fails ($p = 0.334$). Thus the new study confirms the primary quality contrast but does not establish mapped endpoint equivalence. Failure to establish equivalence does not establish nonequivalence.

Padding study. Five seeds on each of two corpora, Wikipedia and FineWeb-Edu. Under fixed padding, the MLM-on and MLM-off runs coincide at every saved checkpoint and in every score (cf. Appendix C), so the study compares a run with the objective restored against one without it (Figure F.2). On Wikipedia, restoring MLM changes transfer accuracy by $+0.46$ points ($[0.08, 0.83]$, 97.5%), the preregistered primary, which did not clear its 0.25-point threshold, and endpoint STS

by -13.1 ($[-14.7, -11.5]$), a secondary contrast. On FineWeb-Edu the transfer effect is $+1.32$ ($[0.98, 1.66]$), reported descriptively under that corpus's plan. The direction of both effects matches what the original authors report (cf. Appendix D in Gao et al., 2021).

F.2 MAMBA-2

A 134M-parameter Mamba-2 language model trained on 100M tokens, five seeds per arm, on the affected two-way tensor-parallel path, the repaired path, a healthy fused path, and an explicit eight-group model on one GPU. Under a common healthy evaluator, test perplexity is 76.48 for the affected model, 70.83 after repair, 70.86 for the healthy path, and 69.05 with internal normalization disabled. The preregistered primary contrast, affected minus repaired test NLL, is 0.0766 nats ($[0.0656, 0.0877]$, 97.5%), clearing the 0.02-nat threshold; repaired minus healthy is -0.0003 ($[-0.0008, 0.0002]$), inside the ± 0.005 recovery margin.

Table F.2 crosses the training rule with the evaluation rule (Figure F.3). Evaluated with eight groups, the affected model scores 4.2826 and the explicit eight-group model 4.2822, a difference of -0.0004 nats whose interval spans zero; no equivalence margin was set in advance. A post-hoc decomposition splits the primary gap, $0.0766 = 0.0544 + 0.0222$ nats: the first term is the cost of evaluating eight-group weights with four groups, the second the difference between the models each under its own rule. The shares, 71% and 29% (joint seed bootstrap 68–74% and 26–32%), describe this study only, and the second term is not purely optimizer damage.

Table F.2: Mamba-2 test NLL (nats) by training and evaluation rule, means over five seeds.

Training rule	Evaluated with 4 groups	Evaluated with 8 groups
Repaired, 4 groups	4.2603	4.3748
Affected, 8 groups executed	4.3370	4.2826
Explicit 8 groups (B/C grouping 4)	4.3355	4.2822

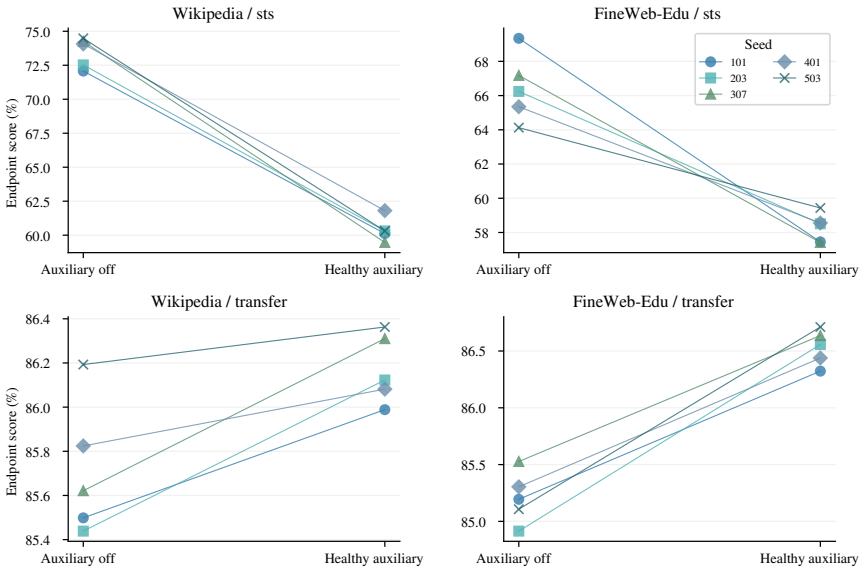

Figure F.2: Padding study: paired five-seed endpoints with the auxiliary objective off (the affected configuration) and restored, on two corpora. Endpoint plots do not turn the unmet Wikipedia primary threshold into a success.

Four-way confirmation. A new preregistered study adds 20 runs: a one-GPU four-group anchor, four-GPU affected and repaired runs, and a one-GPU explicit 16-group mimic, each with the same five paired seeds, 134M parameters, and 100M tokens. All three primary gates pass (Table F.3). Under common four-group evaluation, affected and repaired perplexities are 85.55 and 70.81. Under their respective training rules, their NLL difference is 0.0527 ($[0.0488, 0.0566]$, descriptive); changing the affected checkpoint's evaluation from 16 to four groups adds 0.1364 nats, giving $0.1891 = 0.1364 + 0.0527$. Formal mimic equivalence holds under native 16-group evaluation; under common four-group evaluation, its descriptive interval $[-0.0028, 0.0092]$ extends beyond ± 0.005.

Table F.3: New Mamba-2 four-way confirmation, five paired seeds and 100M training tokens. NLL differences in nats; all three preregistered gates pass.

Contrast	Eval. groups	Mean [CI]	Gate
TP4 affected − repaired	4	0.1891 $[0.1697, 0.2086]$ (97.5%)	lower > 0.02
TP1 mimic − TP4 affected	16	0.0014 $[-0.0010, 0.0038]$ (95%)	within ± 0.005
TP4 repaired − TP1 anchor	4	-0.0007 $[-0.0015, 0.0001]$ (95%)	within ± 0.005

The common four-group development-loss gap remains positive at all 16 scheduled checkpoints, from 0.0583 at 6.55M tokens to 0.1863 at 100M. All five affected endpoints pass TP4-to-TP1 16-group bridge checks. Historical TP2 points use matching data, core sources, and training recipes, but the CUDA-extension binaries were not verified identical; they are secondary evidence, and the new primary tests use only new runs. The results establish persistence through the measured 100M-token window, not convergence at longer budgets.

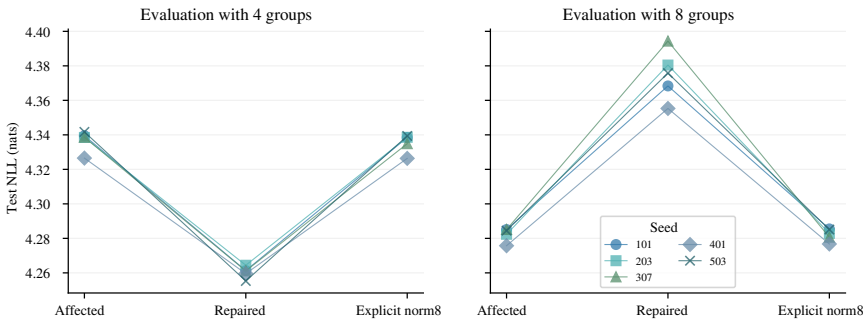

Figure F.3: Earlier two-way Mamba-2 paired cross-evaluation per seed. The same five seeds are reused across evaluation rules and are not independent cells.

F.3 Barlow Twins

ResNet-50 on ImageNet at 224 pixels with the 8192-8192-8192 projector and a batch of 256, three seeds, 2,048 steps, all within warm-up; forward pass and loss in FP64, master weights, gradients, LARS, and buffers in FP32. Five roles were run at one, two, and four GPUs: raw, repaired, mapped (repaired communication with η/W and $W\mu$), and repairs of only the excluded or only the adapted groups. All 45 trajectories, endpoint records, and 90 backbone/projector readouts are complete. The fixed readout uses 100,000 training images and 25,000 validation images from the discovery pool. The mapped and raw readouts agree to recorded precision in all six multi-GPU seed cells. Backbone repair gains are $+1.132$, $+0.900$, $+1.020$ points at two GPUs (mean $+1.017$), and $+1.740$, $+1.588$, $+1.448$ at four (mean $+1.592$). Four-GPU repaired runs remain within 0.056 points of one GPU. Repairing only excluded groups recovers $+1.748$, $+1.640$, $+1.404$ points; repairing only adapted groups gives 0, $+0.028$, -0.052 (Figure F.5). Independent global-reference checks now pass at both two and four GPUs (15/15 each). Results remain exploratory: three seeds do not complete the original five-seed confirmatory design, and all steps precede the 50,040-step warm-up endpoint.

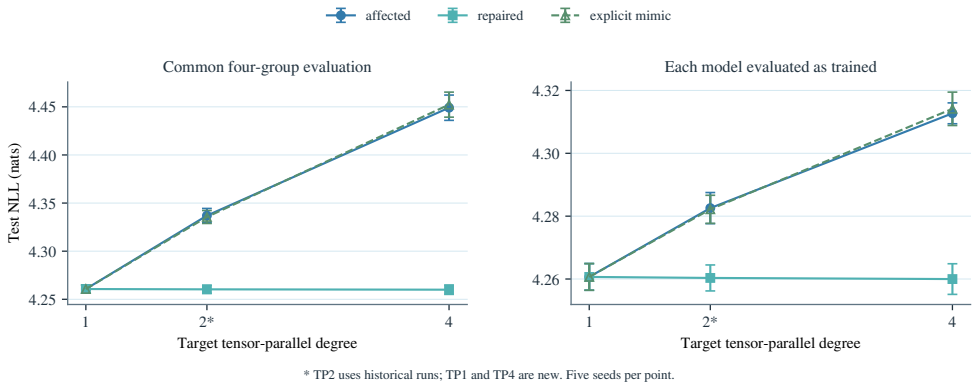

Figure F.4: Mamba-2 at 134M parameters and 100M tokens: test NLL by target tensor-parallel degree, under common four-group evaluation (left) and each model's training rule (right). Explicit mimics run on one GPU. TP2 is historical; TP1 and TP4 are new. Error bars are 95% intervals over five seeds; primary tests use paired differences.

Cross-layout continuation. From fixed step-2,048 checkpoints with nonzero momentum, a separately frozen test transfers the same state and data stream between two and four GPUs. Full hyperparameter and optimizer-state mapping passes the first-step relative-L_2 cap of 0.002 in both directions (maximum errors 1.03×10^{-5} and 9.63×10^{-6}), but fails at step 16 (0.01324 and 0.00512). Same-layout 16-step replay has zero full-state error. The one-step result supports the state-aware mapping; strict cross-layout trajectory equivalence is not established.

Post-warm-up continuation. For each of two seeds, raw, repaired, and mapped runs were branched from that seed's step-50,040 raw checkpoint and continued for 5,000 updates. Repair-minus-raw backbone changes were -1.264 and -0.808 points; mapped and raw endpoint parameters and readouts matched. Independent-reference single-update checks passed at both branching states. This is a late intervention after shared raw training, not a from-initialization comparison.

F.4 MASK2FORMER

Continuation of an official R50 checkpoint for 10,000 steps, two seeds, under a learning-rate schedule frozen before evaluation, with a raw two-GPU arm, a repaired two-GPU arm, and a one-GPU replay of the predicted weights (Table F.4, Figure F.6). The replay equals the raw arm exactly at every checkpoint. Raw minus repaired PQ is $+0.022$ and -0.033 at 10,000 steps (mean -0.005), short of the prospective reference of at least one PQ point in the same direction for both seeds. The reweighting still changes predictions: about 3.3% and 2.9% of predicted pixels change class, in 2,497 of 2,500 images, and class-level gains and losses nearly cancel in the macro-averaged PQ. Image-bootstrap intervals for fixed model pairs ($[-0.17, +0.22]$ and $[-0.24, +0.21]$) are conditional on the trained models and are not seed intervals.

Table F.4: Mask2Former PQ during the continuation (common start 51.313).

Seed	Step	Raw	Repaired	Replay	Raw − repaired
72019	2,000	51.160	51.263	51.160	−0.103
72019	5,000	51.309	51.324	51.309	−0.015
72019	10,000	51.354	51.332	51.354	+0.022
72031	2,000	51.099	51.153	51.099	−0.054
72031	5,000	51.272	51.304	51.272	−0.032
72031	10,000	51.191	51.224	51.191	−0.033

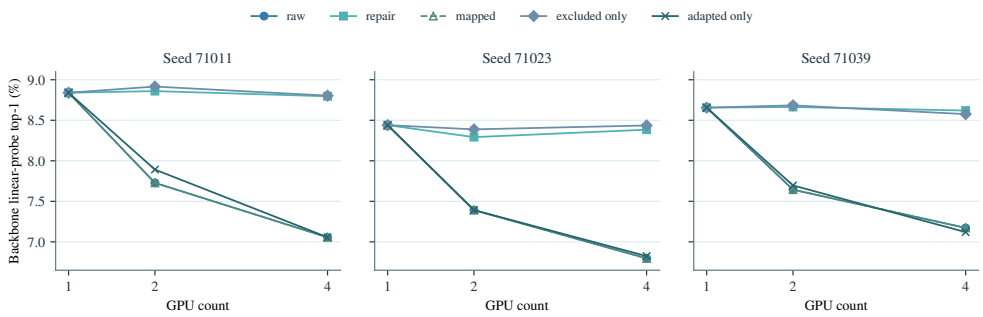

Figure F.5: Barlow Twins, exploratory: backbone linear-probe top-1 by GPU count and role for three seeds (2,048 steps within warm-up).

F.5 GaLore, Q-GaLore, and APOLLO

Quality. A 60M-parameter LLaMA on C4 for 2,000 logical updates, three seeds, comparing each released configuration with its declared update within its own family (Figure F.7). Validation NLL moves from 4.938 to 4.727 for GaLore and from 8.959 to 4.841 for Q-GaLore, of which the float-hook repair alone accounts for a move to 6.073; for APOLLO it rises from 4.889 to 5.046. The comparison is descriptive; configurations differ across families, and no ranking of methods follows.

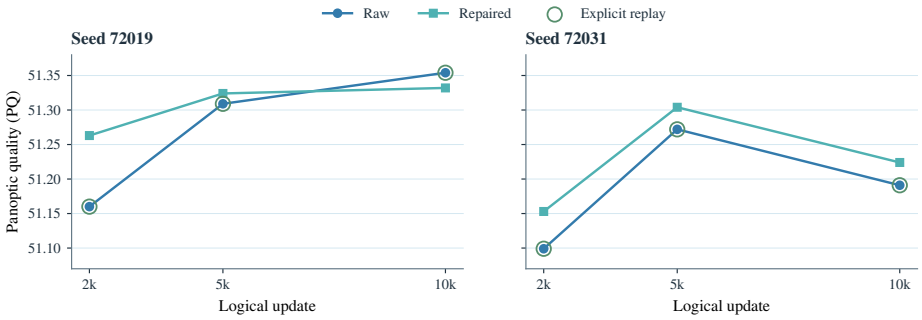

Figure F.6: Mask2Former continuation trajectories (two seeds). Replay equality is separate from the small PQ effects.

Resources. For the README 7B configuration, two 501-update windows on 96 GB cards compare the declared update with GPU-resident gradients (peak allocated 32.49 GiB, 238.5 minutes) and with gradients staged in host memory (20.18 GiB, 290.3 minutes, with a 13.0 GiB host gradient buffer): 37.9% less allocated memory for 21.7% more time (Figure F.8). The released configuration peaks at 20.0 GiB in a separate four-update window. The windows cycle a fixed profiling set and measure resources, not quality or convergence. On a physical 24 GB RTX 3090, tested over the first four updates only, the released configuration and the host-staged declared update complete while the GPU-resident declared update exceeds memory.

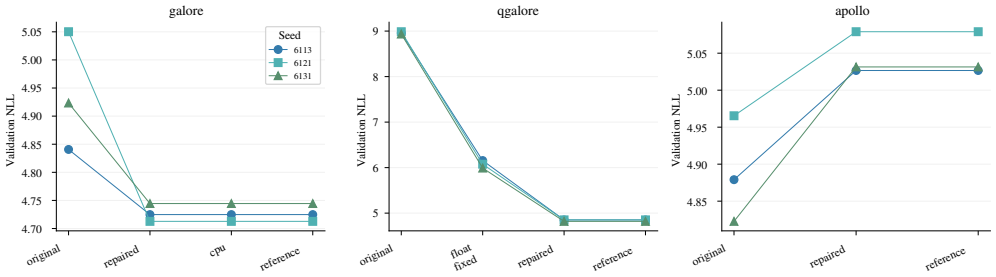

Figure F.7: Within-family validation NLL for the released and declared configurations (three seeds, 2,000 updates). APOLLO's deterioration after repair is retained.

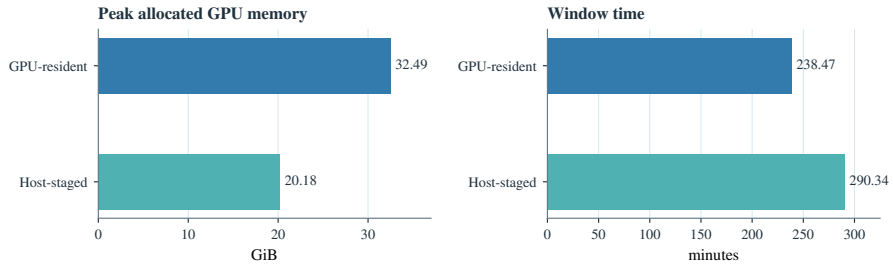

Figure F.8: GaLore 7B resource windows (501 updates each): peak allocated GPU memory and window time for the declared update with GPU-resident or host-staged gradients.

F.6 Prospective predictions and natural-task consequences

Prospective predictions. The initial predictions were blind to experimental outcomes: we inspected the code, froze the predictions and test criteria, and only then ran the experiments. Candidate selection was based on code inspection, so this study does not estimate discovery rates in randomly sampled codebases. Controlled CPU FP64 tests supported all six: relative loss scales in FlagEmbedding and GritLM; the update boundary and EMA clock in Diffusion Policy; the gradient scale and corresponding Adam mapping in StyleGAN2; and omitted CLIP key-side gradients in SLIP.

Scoped repair closure. Subsequent dual-host checks established repair closure for FlagEmbedding's cross-device `kl_div` branch, GritLM's standard $W = 1, 2$ path, SLIP's CLIP backward path, and a separately preregistered Diffusion Policy profile. GritLM's qualification does not cover $W = 4$, GradCache, or DeepSpeed. Diffusion Policy's formal closure is limited to atomic8 v2: a fixed 32-example logical window, physical forward batches of eight, and accumulation $k = 1, 4$. Its original vectorized profile failed numerical qualification; the atomic8 v2 result does not qualify that profile or transfer to the earlier quality results below. StyleGAN2 did not pass numerical qualification and is excluded from the 43-error inventory. The four additions have no new mechanism-group assignments.

Natural-task consequences. A separate preregistration fixed five primary quality comparisons, each using three paired training seeds. Table F.5 reports every primary result as repair minus native. FlagEmbedding improves SciFact retrieval, while GritLM's primary WikiText-2 NLL worsens; the other three primary tests do not pass the prespecified multiplicity correction. The StyleGAN2 quality study fixes the physical forward batch at eight and remains distinct from the failed numerical qualification.

Table F.5: R4 natural-task quality results: repair minus native, three paired training seeds per case, in native metric units. Arrows indicate the preferred metric direction. The prespecified Bonferroni threshold for the five primary tests is $p < 0.01$. Reported intervals are individual 95% paired confidence intervals, not simultaneous confidence intervals.

Case / primary metric	Difference [95% CI]	Five-test correction
FlagEmbedding / SciFact nDCG@10 ↑	$+0.01945$ $[0.01345, 0.02544]$	Passed; repair improves
GritLM / WikiText-2 NLL ↓	$+0.00294$ $[0.00242, 0.00346]$	Passed; repair worsens
Diffusion Policy / PushT mean score ↑	$+0.01755$ $[-0.03497, 0.07006]$	Not passed
StyleGAN2 / FID50k ↓, forward batch 8	$+10.49038$ $[-6.57039, 27.55114]$	Not passed
SLIP / COCO mean bidirectional R@1 ↑	$+0.00313$ $[-0.00237, 0.00863]$	Not passed

The natural-task endpoint replay checks met their preregistered equivalence margins for FlagEmbedding and GritLM. Equivalence was not established for Diffusion Policy's repair-versus-declared comparison or the native-versus-mapped comparisons in StyleGAN2 and SLIP. These outcomes and

the original qualification failures remain part of the evidence; failure to establish equivalence does not establish nonequivalence.

F.7 Published and Documented Configurations

A configuration written down without its resources does not fix the algorithm. The two Barlow Twins setups of §1, 32 GPUs in the paper and 16 in the repository, execute $1/32$ and $1/16$ of the configured learning rate for biases and normalization parameters, and 32 and 16 times the configured weight decay elsewhere. We do not know which setup produced the reported results and make no claim that they would change.

GaLore's README command for single-GPU 7B pre-training runs 32 optimizer steps for every update it declares, and its outer clip never sees a gradient. On 96 GB cards the released configuration peaks at 20.0 GiB, and the declared update at 32.5 GiB with GPU-resident gradients or 20.2 GiB with gradients staged in host memory, at 21.7% more time. On a physical 24 GB RTX 3090, over the first four updates, the released configuration and the host-staged declared update fit, but the GPU-resident declared update exhausts memory (cf. Appendix F.5). The accumulation behavior has been discussed publicly; we evaluate no published memory figure.

A Mamba-2 checkpoint trained on the affected two-way path scores 4.283 nats with the grouping it was trained with but 4.337 when loaded on one GPU with the declared grouping. Recording the executed grouping, reconstructed from the sealed configuration, restores 4.283 (cf. Appendix C).

G Reproducibility and Artifacts

G.1 What each record contains

Every result in the paper is backed by a sealed record rather than a log. A case record binds the audited source by commit or version, the dependency lock, the adapter descriptor and support profile, the frozen protocol, the calibration controls and thresholds, and the raw traces of every arm, including the repair, the negative control, and the oracle. Each file is listed in a manifest with its SHA-256 digest, and a verifier recomputes the digests and the verdicts from the raw traces on a CPU-only host from a clean checkout. A verification receipt establishes self-consistency and detects corruption; it does not authenticate who produced a record. Certificates exist only for the release lanes (cf. Appendix E); the other records carry the stated validation levels (cf. Appendix A.4), and no receipt is extended beyond the record and support profile it binds.

G.2 Environments

There is no single global environment: each record carries its own environment capture. Most current records use PyTorch 2.13 builds, on hosts with one or two NVIDIA RTX 5090 cards for the checker lanes and most single-update qualifications, 96 GB cards for the GaLore resource windows, and a physical 24 GB RTX 3090 for the four-update feasibility test; the new Mamba-2 four-way study uses PyTorch 2.8.0 with CUDA 12.8 on three four-GPU RTX PRO 6000 hosts; theory fixtures run on CPU in FP64. Each receipt records the precision and determinism settings it used, for example deterministic algorithms with TF32 disabled in the checker lanes, and results are claimed only within the precision stated for each record.

G.3 Data, models, and seeds

Datasets and checkpoints are bound by identifier, revision, and digest, and are used under their licenses: Wikipedia and FineWeb-Edu sentences for SimCSE, ImageNet for Barlow Twins, COCO panoptic and an official R50 checkpoint for Mask2Former, and C4 for the GaLore family; synthetic tasks, such as the Qwen3-MoE carrier and the tiny adapter, are generated from recorded seeds. Seeds are part of every record: 101, 203, 307, 401, and 503 for the SimCSE padding study and for Mamba-2; 631, 641, 647, 653, and 659 for the new SimCSE layout confirmation; 71011, 71023, and 71039 for Barlow Twins; 72019 and 72031 for Mask2Former; 1729 for the theory fixtures and 20260924 for the numerical check of Proposition 2; and a separate calibration and formal seed for each real case.

G.4 ARTIFACT

We will release the adapters, calibration records, fault corpus, case records with their verifier, and the scripts that rebuild every appendix table and figure from the sealed records without a GPU. Model checkpoints, training corpora, and large tensors are referenced by identifier and digest rather than included. The released artifact will be anonymized for review; an earlier artifact certification applied only to the artifact it was bound to and is not carried over to the current records.