ALIGNDIAG: STAGE-AWARE MULTI-SOURCE DIAGNOSIS FOR CROSS-HARDWARE TRAINING MIGRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model training increasingly spans accelerator generations and vendors as model development outpaces datacenter renewal. Common frameworks make such migration practical, yet backend-specific implementations can introduce legitimate numerical differences under finite-precision arithmetic. Meanwhile, similar losses may conceal internal faults whose effects propagate through subsequent computations. Reliable diagnosis must distinguish permissible drift from material errors and separate their sources from downstream effects. We introduce AlignDiag, the first staged multi-source diagnostic framework for training migration. It builds a contract-grounded evidence library that distinguishes noise, permitted drift, and controlled faults while preserving their provenance and lineage. Building on this structured evidence, a label-efficient procedure selects role-aware rules without using evaluated metrics to define labels. With these rules fixed, AlignDiag then detects and localizes material divergence automatically and uses targeted replay to verify whether identified candidates are genuine sources of observed deviations. Across our experiments, we organize 5,861 comparison pairs into 844 controlled groups covering expected variation and injected faults. Overall, AlignDiag achieves 98.61% accuracy distinguishing drift from faults. In prospective evaluations spanning training windows from 2 to 1,000 steps across GPU and NPU platforms, AlignDiag detects faults missed by loss-only checks, recovers injected source sets, and exposes sustained routing divergence in MoE models despite near-zero training loss. Multi-source localization reduces scans by 36.0% over iterative single-source strategies while preserving complete recovery.

1 INTRODUCTION

Large language models (LLMs) are rapidly expanding the capabilities and applications of artificial intelligence, driving continued innovation in model architectures and computing infrastructure. Recent models such as GPT-6 Astra (OpenAI, 2026), Claude Opus 5.5 (Anthropic, 2026), and Qwen3.8-Flash-Next (Qwen Team, 2026) illustrate the rapid evolution of model capabilities and computational efficiency. The hardware landscape is similarly diverse, spanning accelerator generations and vendors with different balances of compute throughput, memory capacity, and interconnect bandwidth. Across applications from edge devices to data centers, model requirements must fit available resources, cost constraints, and hardware availability (Lin et al., 2024; Guo et al., 2026). Deploying and adapting models across hardware platforms is therefore an important link between model development and practical use.

Training correctness depends on the underlying hardware stack (Xie et al., 2025). After migration, differences in operator mapping, compute precision, and communication can alter numerical results under finite-precision arithmetic. These differences do not necessarily indicate training errors, but they complicate verification that migration preserves the intended training semantics (Zhan et al., 2025; Lu et al., 2025).

Related work follows two main directions. The first checks consistency across execution configurations to verify the semantic preservation of computation graphs across backends, compilation settings, and optimizations (Pham et al., 2019; Liu et al., 2023a). However, such approaches can conflate numerical inconsistency with semantic errors, reporting valid optimizations as compiler

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.

Table 1: Comparison of existing systems and AlignDiag across target settings, comparison metrics, and diagnostic capabilities. $\theta^\star$ denotes calibrated thresholds.

System	Target	Metric	Cross hardware	Localization	Numerical tolerances	Downstream recovery
Ekka (Gu et al., 2026)	Cross-graph	Normalized L2	×	First module	✓	×
OpGuard (Zhou et al., 2026)	Cross-parallelism	Bitwise	✓	First operator	×	×
TTrace (Jiang et al., 2025a)	Cross-parallelism	Rel. Frobenius error	×	Multi source	✓	×
msProbe (Ascend, 2026)	Cross-framework	Cos, abs/relative error	Partial	First module	✓	×
DriftBench (Vitale, 2026)	Cross-migration	Output flip rate	✓	×	×	×
AlignDiag (Ours)	**Cross-migration**	**Role-aware + $\theta^\star$**	✓	**Multi source**	✓	✓

bugs. The second checks and diagnoses training execution, focusing on the correctness of gradients, parameter updates, and training state under distributed partitioning, reductions, and synchronization (Lu et al., 2025; Jiang et al., 2025a; Zhou et al., 2026). Yet these approaches operate within single iterations and may miss accumulated errors across steps. Bridging configuration comparison and correctness assessment, DriftBench predicts serving-environment migration risk from task-output drift (Vitale, 2026). But its core assessment remains black-box, without localizing internal errors, tracing their propagation, tracking training deviations, or validating repairs. The comparison is summarized in Table 1.

Our goal is to establish criteria for cross-hardware training alignment that allow numerical variation, and to connect these decisions to internal error localization and validation of multiple error sources. We also aim to diagnose problems with minimal labels and short training windows, saving resources and detecting errors early. This requires addressing three key challenges.

Reliable validation and diagnosis face three challenges. First, bitwise differences need not imply faults, while similar losses can conceal internal errors, making it difficult to distinguish numerical variation from errors through these comparisons alone. Second, establishing reliable decision rules requires accounting for the different numerical properties of probabilities, gradients, and optimizer states. These differences limit the applicability of a single threshold and make systematic validation of state-specific criteria necessary. Finally, error propagation and interactions among multiple sources complicate attribution even after an error is detected. Upstream deviations propagate downstream, and multiple sources can mask one another's effects, so a local correction alone does not establish the root cause.

To address these challenges, we present AlignDiag, a semi-automatic framework for validating and diagnosing cross-hardware training migration through label-efficient learning. The core idea is to use controlled samples to establish alignment criteria for different training states, then use these decisions to guide iterative diagnosis. Local changes and downstream responses to interventions provide evidence about error sources. Specifically, AlignDiag ensures: 1) key internal tensors and training states are compared before and after migration to identify meaningful deviations beyond normal numerical variation; 2) labeled samples are used to select metrics, calibrate thresholds, and combine rules to obtain executable criteria for different training states; 3) misaligned locations are compared under identical inputs, local errors are iteratively neutralized, and remaining deviations are checked, followed by joint interventions and withdrawal of each intervention in turn to test the downstream effects of multiple sources.

For training framework developers and model migration teams, AlignDiag provides internal computation checks and error-source diagnosis during cross-hardware migration. We evaluate AlignDiag on 844 experimental groups and 5,861 comparison pairs across model architectures, hardware stacks, and training stages, achieving forward and backward group-balanced accuracies of 99.07% and 94.17%, with an overall accuracy of 98.61%. Compared with iterative single-source localization and repair, multi-source localization reduces scans by 36.0% when all sources are initially visible, while preserving complete source recovery. These results demonstrate accurate detection and efficient, complete source localization for migration debugging.

In summary, our contributions are:

1. We build a controlled evidence library for cross-hardware training validation, with independent labels and grouping of samples that share a common origin to support rule calibration and evaluation.

2. We introduce a semi-automatic method that selects and calibrates metrics and decision rules for different training states, then validates and freezes the rules before evaluation.
3. We design a diagnostic procedure that detects training deviations and localizes multiple error sources through comparisons under identical inputs, iterative neutralization, joint interventions, and withdrawal of individual interventions.

2 Background and Motivation

Cross-hardware migration introduces numerical variation from implementation and precision differences, requiring criteria that distinguish permitted variation from real errors.

2.1 Numerical Variation and Alignment Assessment

Hardware migration can change numerical results without changing intended computation. Different backends may evaluate reductions in different orders or round intermediate results at different points; with finite precision, these choices change outputs even when the mathematical operation is unchanged (Goldberg, 1991; PyTorch Contributors, 2026a). Existing bitwise checks (Zhou et al., 2026) are too strict, while tolerance-based approaches (Wang et al., 2025; Jiang et al., 2025a) may miss errors.

Loss alone cannot establish correctness: spatially, averaging dilutes changes ($\Delta L = B^{-1} \sum_i \Delta \ell_i$); temporally, loss precedes backward and optimizer steps (PyTorch Contributors, 2026b), as shown in Appendix Figure G.1. Production cases confirm that inconsistent updates can remain hidden in loss metrics (Jiang et al., 2025b; Zhou et al., 2026).

Reliable assessment requires complementary measures with calibrated thresholds. Single measures provide partial views; combining metrics across training stages offers more complete assessment (Klabunde et al., 2025b). Calibrated dtype- and operator-specific tolerances recover missed bugs while managing false alarms (Budzinskiy et al., 2025; Sarkar, 2026). Learning these rules requires samples distinguishing permitted changes from faults, with labels independent of evaluated metrics (Jiang et al., 2025b).

2.2 Error Propagation and Localization

Locating training errors is harder than detecting them. Backends may fuse, split, or distribute operations to suit target hardware, so operator calls no longer match one-to-one. Localization therefore needs common semantic boundaries, not identical execution order (Lu et al., 2025; Zhou et al., 2026).

Additionally, errors may be detected later than introduced (Jiang et al., 2025a; Zhou et al., 2026). An incorrect gradient produced during backward computation may be detected only when updated parameters are checked; restoring affected parameters removes downstream symptoms without addressing the source (Guo et al., 2020b; Cleve & Zeller, 2005). An anomaly shows where a problem becomes visible; a successful restoration shows where effects can be blocked; neither alone establishes the true origin. Reliable localization must match corresponding computations and distinguish error sources from their later effects. For example, a Softmax mismatch following an RMSNorm fault may arise from error propagation or an additional local fault; identifying the upstream source alone cannot distinguish these cases.

3 Overview

AlignDiag compares reference and migrated training executions under controlled conditions and connects alignment assessment to source diagnosis (Figure 1). The evidence library and frozen rules at the top support two complementary assessment dimensions—forward and backward—that validate migration correctness from different perspectives. Forward assessment examines numerical consistency of outputs and activations, while backward assessment checks gradient correctness and optimizer state evolution. Violating states and their dependency context feed the localization, recovery, and confirmation procedures on the right.

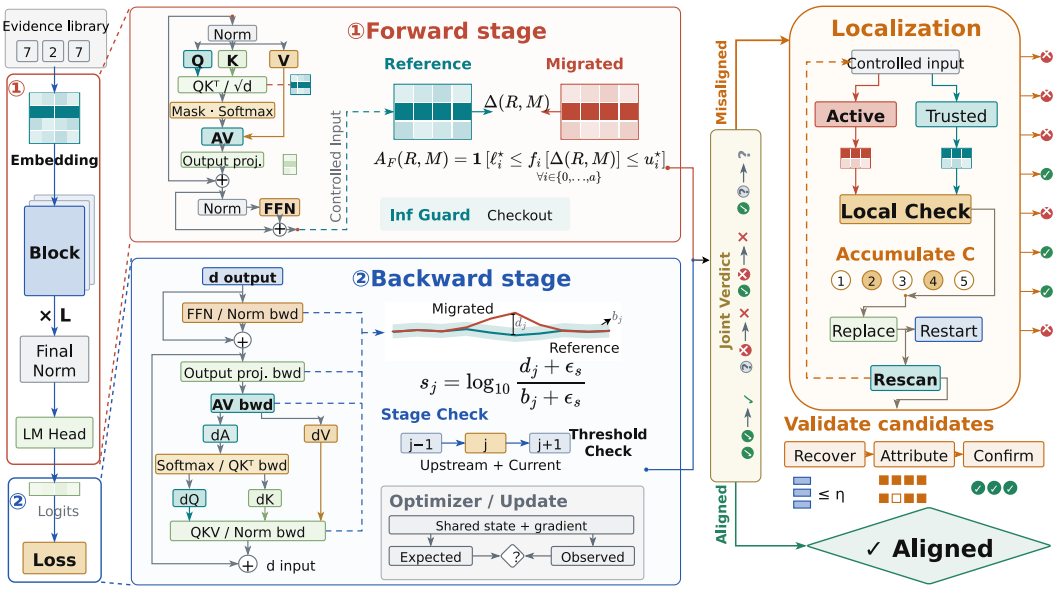

Figure 1: **AlignDiag overview.** Controlled evidence supports frozen rules for forward, backward, and optimizer checks. Violations guide same-input localization, cumulative neutralization and rescanning, exact-set recovery, contribution analysis, and fresh paired confirmation. Optimizer checks share the same pre-update conditions. Curves and status combinations are schematic.

Task-level benchmarks and test outputs alone do not establish whether an observed numerical discrepancy reflects permitted variation or a training fault. Calibrating this distinction requires controlled comparisons with independently justified labels, motivating our evidence library. The evidence library organizes reference repeats, permitted implementation changes, and controlled faults into interpretable comparisons. Labels distinguish permitted variation from violations using evidence independent of the metrics under evaluation. Records with shared origins are grouped before data splitting, preventing related observations from crossing calibration, validation, and test partitions. The resulting rules and thresholds are frozen after independent validation.

Forward assessment examines numerical consistency through exact comparisons of outputs and activations from matched executions; Backward assessment validates gradient correctness using scale-normalized comparisons across registered stages, including optimizer state checks against declared equations. Together, these two dimensions provide complementary evidence of training correctness, ensuring that agreement in one direction does not mask violations in the other. Accordingly, AlignDiag declares misalignment whenever a violation is confirmed and establishes alignment only when all required checks pass.

When misalignment is detected, localization identifies error sources through iterative intervention at registered boundaries. Active computations are replaced with trusted references cumulatively, with remaining sites rescanned to expose masked sources. Recovery validates whether the identified set restores required endpoints, while fresh paired executions confirm repeat-stable results.

4 System Design

4.1 State-Aware Alignment Metrics

To assess migration correctness, we distinguish acceptable numerical variation from real errors that disrupt training. Hardware migration introduces kernel-induced numerical variation (PyTorch Contributors, 2026a), while misalignment errors may arise in operators, gradient synchronization, or optimizer execution, affecting later states and parameter updates (Jiang et al., 2025a;b; Micikevicius et al., 2018). These varied sources call for complementary checks across training stages.

AlignDiag organizes metrics around three aspects of a training update. Numerical Value Consistency uses exact and tolerance-based elementwise comparisons, maximum absolute difference, relative L_2 distance, and infinity counts $(E, A, D_{\max}, D_{\mathrm{rel}}, C_\infty)$; Optimizer State Consistency checks first-

and second-moment states against their expected values using residuals ($m_t^\star, v_t^\star, R_m, R_v$); Update Behavior Consistency uses expected parameter increments to assess sign disagreements and lost updates ($B_{\text{sign}}, B_{\text{lost}}$). Together, these metrics assess numerical precision, optimizer mechanics, and update application. Detailed definitions appear in Appendix A.

4.2 Semi-Automated Rule Construction

AlignDiag evaluates individual metrics and their combinations on well-labeled samples organized into dependency-audited groups, guiding the metric selection in Section 4.1. With the metric inventory and rule structure fixed, automatic calibration fits state-specific thresholds to balance false alarms and missed faults. Matched controls provide tolerance estimates for checks based on normal variation. Thresholds are frozen after meeting predefined criteria on independent validation groups.

At runtime, AlignDiag applies its frozen rules through two complementary assessment dimensions: forward and backward. Each dimension returns its decision, coverage, and supporting observations. These results jointly determine the overall verdict: a confirmed violation of any required check establishes misalignment, while alignment requires complete coverage and agreement across all required checks from both dimensions. As a safeguard, missing required evidence leaves the result undecided in the absence of a confirmed violation.

Forward assessment. Forward checks apply exact numerical comparisons to states selected by the frozen evidence mapping. For passing cases, the supplementary branch checks infinity counts against calibrated thresholds when enabled. The forward violation indicator is

$$v_{\text{F}} = b_{\text{F}} \vee (c_{\text{F}} \wedge \neg a_{\text{F}}), \tag{1}$$

where b_{F} indicates a violation established by the base checks, c_{F} indicates an alert triggered by the supplementary count check, and a_{F} indicates that all registered tolerance checks pass. The tolerance checks can therefore suppress a supplementary alert without overturning a base violation.

Backward assessment. Backward checks measure stage-wise deviations relative to matched controls. For stage j, the deviation score is

$$s_j = \log_{10} \frac{D_{\max}(x_j, y_j) + \epsilon_s}{\max_{c \in \mathcal{C}_j} D_{\max}(x_j, c_j) + \epsilon_s}, \tag{2}$$

where x_j and y_j are the matched reference and migrated tensors, $\mathcal{C}_j$ is the matched-control pool, and ϵ_s is a numerical stabilizer. AlignDiag jointly examines the current gradient stage and its registered upstream stages, reporting misalignment if any valid score exceeds the frozen boundary. Triggering stages and observations are retained for subsequent diagnosis.

Given the supplied gradient and pre-update states, AlignDiag additionally computes expected optimizer states and parameter increment from the declared update equations, checking moment residuals, update-sign disagreement, and lost updates. Once all four optimizer and update checks have complete, valid evidence, any threshold violation establishes misalignment for this module.

AlignDiag returns a single alignment verdict with supporting states, checks, and observations, and passes confirmed violations with their dependency context to source diagnosis to investigate discrepancy origins and propagation. Appendix B provides the calibration procedure, complete decision rules, and module-level flowcharts.

4.3 Alignment Assessment and Source Diagnosis

AlignDiag localizes multiple error sources at registered training boundaries and uses controlled interventions to validate their effects. The process involves two stages.

Iterative localization alternates cumulative neutralization with rescanning, starting from an empty candidate set $\mathcal{S}_0$. Each round restarts from the same initial state and replaces computations in $\mathcal{S}_k$ with trusted counterparts on their current inputs. At the remaining registered sites, it compares active and trusted outputs on identical inputs and updates the candidate set by

$$\mathcal{S}_{k+1} = \mathcal{S}_k \cup \left\{ j \in \mathcal{J} \setminus \mathcal{S}_k \;\middle|\; v_j^{(k)} = 1,\ \delta_j^{(k)} > \tau_j \right\}, \tag{3}$$

where $\mathcal{J}$ contains the registered sites, $v_j^{(k)} = 1$ requires valid measurements from all expected ranks, and $\delta_j^{(k)}$ is their maximum relative-L_2 discrepancy against trusted outputs, compared with the frozen tolerance τ_j. The next round neutralizes the expanded set together and rescans the remaining sites. The resulting changes in downstream inputs can expose previously masked sources. A complete scan with no new candidates ends the search; missing evidence or budget exhaustion leaves a partial result. Responses are reused for the same mask and execution context, with additional runs limited by the query plan.

Building on these candidates, AlignDiag uses intervention-based validation to examine their effects. AlignDiag first neutralizes the exact candidate set and compares the required outputs, loss, and parameter updates with a matched reference. All required endpoint differences must remain within their frozen tolerances for the set to achieve closure. Individual contributions, in turn, are assessed by neutralizing each site alone and by withdrawing its neutralization while the other registered sites remain neutralized. These comparisons use the same fixed endpoint criteria. Gaps in the contribution evidence trigger further subset analysis within the query budget and available responses. Finally, the candidate set is fixed for fresh paired runs. Recovery is reported as repeat-stable only after every scheduled pair passes the same endpoint checks.

Ultimately, the resulting report links source candidates and their discovery contexts to intervention effects, endpoint recovery, and unresolved checks, providing developers local evidence and replay results for investigating each candidate and tracing targeted changes. Appendix C details the localization and intervention protocols, with derivations and a procedural flowchart.

5 Evaluation

We evaluate AlignDiag's ability to detect training-state errors, localize their sources, and validate their effects through intervention. Section 5.2 presents diagnostic accuracy, source recovery, and analysis cost. Section 5.3 compares with prior work. Section 5.4 analyzes component contributions and calibration efficiency. Section 5.5 examines cross-configuration generalization and case studies.

5.1 Implementation and Experimental Setup

AlignDiag is implemented in Python on PyTorch with backend-specific adapters, using hooks at module boundaries and optimizer steps. The evidence library contains 5,861 pairs across 844 groups from CPUs, NVIDIA H100/H200 GPUs, and Ascend 910B2C NPUs, spanning Dense, MoE, and Sparse architectures (Qwen3, Llama, OLMo, Mixtral, GLM, Kimi) across 17 mechanism categories with lineage-isolated data splitting. For prospective evaluation, experiments use H200 GPUs and Ascend 910B2C NPUs with matched inputs (batch 16, sequence 512, 2 steps), exposing four sites at RMSNorm (layers 0, 27) and RoPE/Softmax (layer 14). Matched controls and controlled faults provide labels with frozen configurations.

5.2 Diagnostic Effectiveness and Cost

Detection and source localization. AlignDiag achieves 91.10% forward and 90.17% backward group-balanced accuracy in Table 2. Overall correctness reaches 98.61% accuracy and 91.90% balanced accuracy. Forward and backward assessment provide complementary coverage of outputs, gradients, and parameter updates. On controlled faults at four boundaries in Qwen3-0.6B, AlignDiag exactly recovers all 16 source subsets across 32 groups on both H200 and NPU as Appendix D.2 shows.

Intervention-based explanation and repair. AlignDiag identifies three candidate sources (RMSNorm layer 0, RoPE and Softmax layer 14) whose joint repair eliminates logit, loss, and parameter-update errors on both H200 and NPU, while every proper subset leaves residual mismatches (Figure 2). Partial repairs reveal endpoint-dependent effects: on H200, repairing only RMSNorm reduces logit error to 19.5% while leaving update error at 87.8%; on NPU, it reduces logit error but increases loss error to 3.11$\times$ baseline. These endpoint-dependent responses support joint verification of logits, loss, and parameter updates. In 21 multi-source cases, partial repairs leave residuals and sometimes increase loss error despite reducing logit error; repairing the full predicted set restores all three endpoint classes (Appendix Figure D.5). Across all 37 completed results with nonempty can-

Table 2: **AlignDiag detection performance across training stages.** Accuracy is pair-weighted; recalls and balanced accuracy are group-weighted. Overall reports full-model prediction based on combined assessments.

Training stage	Accuracy (%)	Recall (%)		Balanced accuracy (%)
		Aligned	Misaligned	
Forward	99.07	95.24	86.96	91.10
Backward	94.17	100.00	80.35	90.17
Parameter update	100.00	100.00	100.00	100.00
Overall	98.61	95.24	88.56	91.90

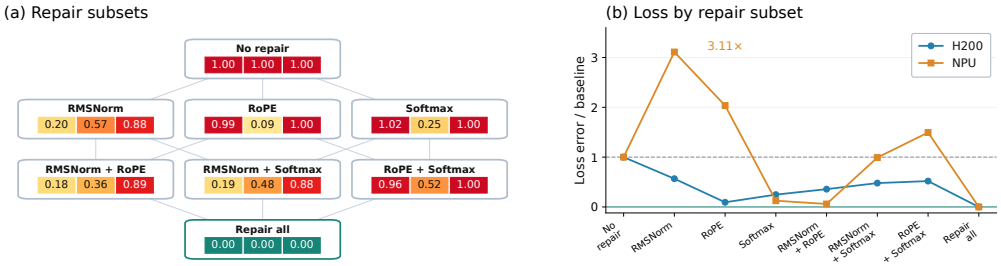

Figure 2: **Repair effects in a three-source case.** (a) Independently executed repair subsets on H200; cells show baseline-normalized logit, loss, and update errors (left to right). RMSNorm is in layer 0; RoPE and Softmax are in layer 14. (b) H200 and NPU loss responses. Joint repair restores all endpoints; partial repair can worsen an endpoint. Details in Appendix D.4.

didate sets, joint repair restores logits, loss, and parameter updates in all repetitions (111 successful runs), while all 77 withdrawal tests reintroduce mismatches.

Longitudinal validation over 1,000 training steps. We extend evaluation to 1,000-step runs on Qwen3-30B-A3B (MoE) and Qwen3-0.6B (Dense) with precision-change interventions. For MoE, localization at step 1 identifies router Softmax; by step 1,000, top-8 expert sets differ at 95.32% of positions—yet both paths achieve near-zero training loss (3.0×10^{-8} and 2.1×10^{-8}), demonstrating loss-only checks cannot detect sustained routing divergence (Appendix Figure F.1). The Dense case shows overlapping training-loss trajectories while validation NLL rises from approximately 3.3 to 7 in both paths, with final NLLs of 7.12 and 7.04. The MoE case connects early source localization with sustained routing divergence, while the Dense case tracks training fit and held-out quality over 1,000 steps (Appendix F).

Offline analysis cost. Full refitting and evaluation take 98.35 s for forward and 99.14 s for backward. Once rules are fixed, evaluation over all 858 forward and 1,114 backward pairs takes a median of 51.00 s and 18.67 s, respectively (Appendix Figures D.3 and D.4).

5.3 Comparison with Prior Work

Error detection. We compare AlignDiag with representative numerical criteria: PyTorch allclose, relative norm error (Jiang et al., 2025a), maximum absolute error, and cosine similarity (Ascend, 2026), fitting their thresholds on training data (Figure 3). In forward assessment, AlignDiag achieves 86.96% fault recall at 4.76% false positives. Shared-threshold baselines reach the same recall with 56.04%–62.64% false positives; AlignDiag reduces false positives by 51–58 percentage points. Cosine similarity reduces false positives to 28.02% but recall falls to 60.87%. In backward assessment, AlignDiag achieves 80.35% recall with zero false positives, while stage-specific baselines reach 42–57% recall at 23–25% false positives.

Source localization. We compare AlignDiag's multi-source search with First-only and One-at-a-time strategies on Qwen3-0.6B/NPU. Both iterative strategies recover all sources in the all-visible and partly masked cases, while First-only recovers the exact set in 6/24 and 0/8 cases, respectively. In all-visible cases, AlignDiag reduces total scans from 75 to 48 (36.0% reduction, 95% CI: 28.4–42.2%). At a two-scan budget, AlignDiag completes all 24 all-visible cases, compared with six

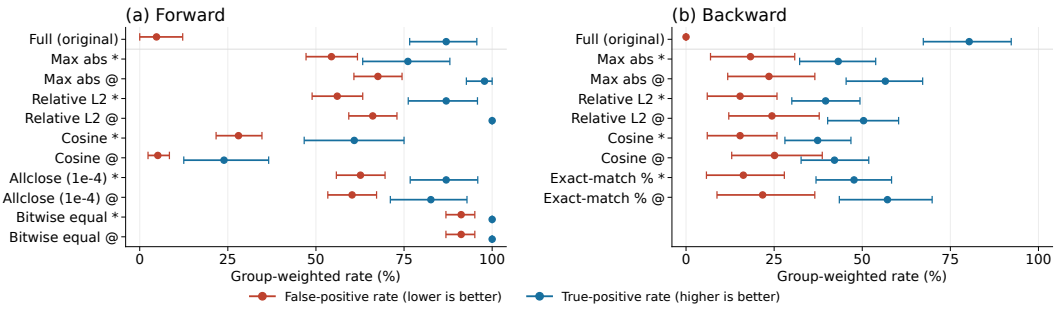

Figure 3: **Numerical criteria and calibration ablations.** (a) Forward; (b) backward. Red/blue points show group-weighted false-positive/fault-recall rates with 95% confidence intervals. All thresholds fitted on training data and frozen for evaluation. Details in Appendix D.5.

for One-at-a-time (Appendix Figure D.6). Across one to four initially visible sources, AlignDiag requires two scans, compared with two to five for One-at-a-time.

5.4 Component Contributions and Calibration Efficiency

Component contributions. Appendix Figure E.3 shows each metric branch's contribution under frozen configurations. In forward assessment, bitwise agreement provides the main backbone (removal reduces balanced accuracy from 90.21% to 20.39%), while count triggers recover additional faults and alignment veto suppresses false alarms. Current-stage and upstream evidence contribute to backward detection. Update checks additionally detect nine fault pairs missed by upstream evidence, increasing balanced accuracy from 88.80% to 90.17%.

Single-metric and calibration ablations. Figure 3 tests whether individual numerical criteria can replace combined decisions. In forward assessment, several criteria achieve high recall but with high false-positive rates (e.g., bitwise equality: 100% recall, 91.21% false positives); AlignDiag achieves 86.96% recall at 4.76% false positives. In backward assessment, the strongest single criterion (stage-specific exact-match proportion) achieves only 67.71% balanced accuracy versus AlignDiag's 90.17%. The effects of finer-grained thresholds vary across metrics, highlighting the need to balance fault sensitivity and numerical tolerance during calibration.

Few-shot forward calibration. Forward threshold calibration reproduces full-data performance using six labeled pairs per fold, with the metric inventory and decision structure fixed. Five-fold evaluation shows accuracy increasing from 85.62% with two pairs to 91.89% with six, with all 20 support draws matching full-data results (Figure 4). A further 100 draws from balanced, evidence-complete support sets reproduce this result (Appendix Figure E.2).

5.5 Cross-configuration Generalization and Diagnostic Case Studies

Generalization across model architectures. To assess transferability beyond Qwen3, we extend evaluation to three additional model families: Llama-3.2-3B, OLMo-3-1025-7B, and Granite-4.0-h-tiny-base, retaining selected metrics and recalibrating numerical parameters on training samples from each model. AlignDiag achieves forward and backward balanced accuracy of 97.86% and 94.64%, with aligned recall of 100% and misaligned recall of 95.73% and 89.29% (Appendix Table 4). These results demonstrate reuse of the selected metrics across the three additional model families through model-specific parameter calibration.

Generalization across target hardware. Extending to NVIDIA RTX 4090 (consumer GPU) using H200 as reference, we adapt metric combinations and parameters through mixed-source learning: 44 repository groups (applicable to both platforms) and 30 RTX-specific groups for calibration, with 50 target-hardware groups held out for validation. Forward and backward balanced accuracy reach 97.78% and 92.89%, with misaligned recall of 100% and 93.33% (Appendix Table 4). These results demonstrate that mixed-source calibration adapts AlignDiag to RTX 4090 by supplementing reusable evidence with target-specific samples.

Representative diagnostic cases. Two precision-change cases (router Softmax in Qwen3-30B-A3B and RMSNorm in Qwen3-0.6B) combine early observations, 1,000-step traces, and final evaluations

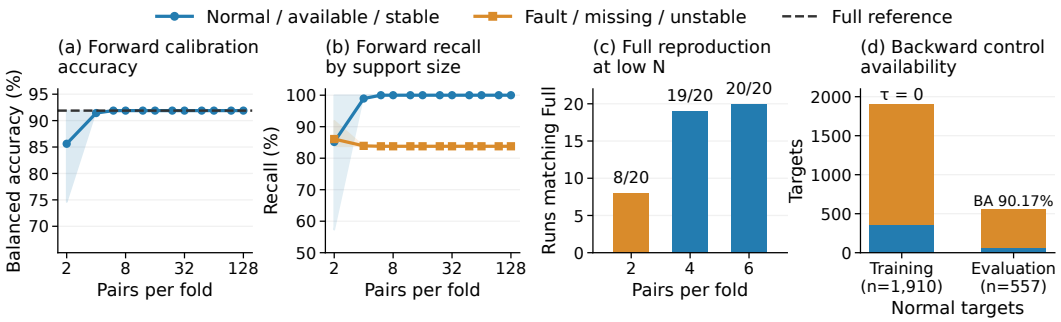

Figure 4: **Few-shot forward calibration.** (a) Group-balanced accuracy; (b) class recalls; (c) draws reproducing full-data performance; (d) backward matched-control availability. Curves show means and shading P10–P90 across support draws. Six pairs per fold reproduce full-data performance in all 20 draws. Appendix E.4.

(Appendix F, Figures F.1 and F.2). As Section 5.2 explains, similarly low training losses can conceal substantially different internal execution paths; operator-level diagnosis exposes these differences.

6 Related Work

Validation and diagnosis methods. Differential and generative testing expose inconsistencies in deep-learning libraries, compilers, and distributed training (Pham et al., 2019; Guo et al., 2020b; Wang et al., 2020; Liu et al., 2023a;b; Yang et al., 2023; Wang et al., 2025), while formal verification reasons about execution-plan equivalence (Lu et al., 2025; Arora et al., 2025). Studies of floating-point arithmetic, mixed precision, and numerical-fault analysis provide foundations for constructing checks (Goldberg, 1991; Micikevicius et al., 2018; Zahid et al., 2024; Badash et al., 2026; Budzinskiy et al., 2025; Kloberdanz et al., 2022; Li et al., 2023; Sharmin et al., 2025; Jiang et al., 2025b; Klabunde et al., 2025a;b; Sarkar, 2026). Cross-stack tools align internal states to diagnose discrepancies (Jiang et al., 2025a; Zhou et al., 2026; Gu et al., 2026; Ascend, 2026), and migration studies examine cross-platform execution drift (Vitale, 2026; Ling Team et al., 2025; Tran & Dang, 2026; Cui et al., 2026; Zheng et al., 2026). AlignDiag calibrates state-specific rules from labeled evidence and connects assessments to intervention-based localization.

Intervention-based source localization. Debugging techniques use controlled changes to investigate error sources (Zeller & Hildebrandt, 2002; Cleve & Zeller, 2005; Sanchez-Stern et al., 2018; Guo et al., 2020a; Miao et al., 2023). TTrace applies same-input rewriting to separate local discrepancies from upstream propagation (Jiang et al., 2025a). AlignDiag combines cumulative neutralization with residual rescanning to expose input-dependent masking, then validates whether the exact candidate set restores required endpoints through withdrawals and fresh paired runs.

7 Conclusion and Discussion

We introduce AlignDiag, a diagnostic framework combining stage-aware alignment assessment with multi-source fault localization for cross-hardware training migration. Calibrated rules distinguish permitted numerical variation from semantic violations, while iterative interventions localize candidate sources and validate their downstream effects. Across the evaluated cases, similar training losses can conceal persistent internal divergence, and partial repairs can improve some endpoints while worsening others. These findings support internal-state assessment and recovery validation through joint interventions, withdrawal tests, and multiple endpoints.

AlignDiag relies on labeled calibration evidence and trusted references, with localization limited by registered observation and intervention boundaries. Our transfer experiments reuse metrics and evidence but still require target-specific calibration and validation. Longitudinal evidence comes from two controlled 1,000-step precision-change cases, while current timings cover offline analysis. Further work should extend diagnostic coverage, reduce target-specific calibration effort, and evaluate longer, more diverse training workloads with instrumentation, reference execution, and physical replay costs included.

AI USE STATEMENT

In this work, we used generative AI tools for assistance with mathematical proofs, debugging method implementations, writing and running programs to generate data according to author-specified rules, cleaning and reformatting datasets, translation, and interpretation of experimental results. We have not used generative AI tools for proposing or refining research hypotheses, developing the conceptual framework, designing the core methodology or experiments, or qualitative and thematic data analysis, and formulating new mathematical claims is not applicable to this work. Additionally, we used generative AI tools for manuscript drafting and language editing, literature organization, submission of experiment scripts, and LaTeX formatting. The figures and plots were created by the authors; no AI-generated images were used. We have reviewed all AI-assisted work. This review covered the AI-assisted proofs, code and data-processing workflows, interpretations of results, and manuscript text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Anthropic. Introducing Claude Opus 5.5. Official model announcement, 2026. URL https://www.anthropic.com/claude-opus-5-5. Accessed September 26, 2026.

Jai Arora, Sirui Lu, Devansh Jain, Tianfan Xu, Farzin Houshmand, Phitchaya Mangpo Phothilimthana, Mohsen Lesani, Praveen Narayanan, Karthik Srinivasa Murthy, Rastislav Bodik, Amit Sabne, and Charith Mendis. TensorRight: Automated verification of tensor graph rewrites. *Proceedings of the ACM on Programming Languages*, 9(POPL):29:1–29:32, 2025. URL https://doi.org/10.1145/3704865.

Ascend. MindStudio Probe: PyTorch accuracy comparison. Official software documentation, 2026. URL https://github.com/Ascend/msprobe/blob/master/docs/zh/user_guide/accuracy_compare/pytorch_accuracy_compare_instruct.md. Accessed September 20, 2026.

Erez Badash, Dan Boneh, Ilan Komargodski, and Megha Srivastava. Hawkeye: Reproducing GPU-level non-determinism. In *Proceedings of Machine Learning and Systems*, volume 8, pp. 1830–1842. MLSys, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/hash/e217c271a57c365a246b0ad39e668ba8-Abstract-Conference.html.

Stanislav Budzinskiy, Wenyi Fang, Longbin Zeng, and Philipp Petersen. Numerical stability analysis of large language models. *arXiv preprint arXiv:2503.10251*, 2025. doi: 10.48550/arXiv.2503.10251. URL https://arxiv.org/abs/2503.10251v2. Version 2, revised June 22, 2026; first version published in 2025.

Holger Cleve and Andreas Zeller. Locating causes of program failures. In *Proceedings of the 27th International Conference on Software Engineering*, pp. 342–351. ACM, 2005. doi: 10.1145/1062455.1062522. URL https://www.st.cs.uni-saarland.de/papers/cz2004/cz2004.pdf.

Weihao Cui, Ji Zhang, Han Zhao, Chao Liu, Jian Sha, Bo Sang, Bingsheng He, Minyi Guo, and Quan Chen. FLARE: Anomaly diagnostics for divergent LLM training in GPU clusters of Thousand-Plus scale. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*, pp. 1021–1035. USENIX Association, 2026. URL https://www.usenix.org/conference/nsdi26/presentation/cui.

David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991. doi: 10.1145/103162.103163. URL https://dl.acm.org/doi/10.1145/103162.103163.

Yile Gu, Zhen Zhang, Shaowei Zhu, Xinwei Fu, Jun Wu, Yida Wang, and Baris Kasikci. Ekka: Automated diagnosis of silent errors in LLM inference. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/66507.

Hui Guo, Ignacio Laguna, and Cindy Rubio-González. pLiner: Isolating lines of floating-point code for compiler-induced variability. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–14. IEEE, 2020a. URL https://doi.org/10.1109/SC41405.2020.00053.

Qianyu Guo, Xiaofei Xie, Yi Li, Xiaoyu Zhang, Yang Liu, Xiaohong Li, and Chao Shen. Audee: Automated testing for deep learning frameworks. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pp. 486–498. ACM, 2020b. URL https://doi.org/10.1145/3324884.3416571.

Runsheng Guo, Utkarsh Anand, Khuzaima Daudjee, and Rathijit Sen. Zorse: Optimizing LLM training efficiency on heterogeneous GPU clusters. In *Proceedings of Machine Learning and Systems*, volume 8, pp. 1569–1582, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/hash/bfa6dd59c1d7f7c785909f9ff7cffe67-Abstract-Conference.html.

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. doi: 10.48550/arXiv.2506.09280. URL https://arxiv.org/abs/2506.09280v2. Version 2, revised January 6, 2026.

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. In *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*, pp. 313–329. USENIX Association, 2025b. URL https://www.usenix.org/conference/osdi25/presentation/jiang.

Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A survey of functional and representational measures. *ACM Computing Surveys*, 57(9):1–52, 2025a. URL https://doi.org/10.1145/3728458. Article 242.

Max Klabunde, Tassilo Wald, Tobias Schumacher, Klaus Maier-Hein, Markus Strohmaier, and Florian Lemmerich. ReSi: A comprehensive benchmark for representational similarity measures. In *International Conference on Learning Representations*, pp. 18701–18740, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/2eef1f75516b0cfd3e944345e5f88c08-Abstract-Conference.html.

Eliska Kloberdanz, Kyle G. Kloberdanz, and Wei Le. DeepStability: A study of unstable numerical methods and their solutions in deep learning. In *Proceedings of the 44th International Conference on Software Engineering*, pp. 586–597. ACM, 2022. URL https://doi.org/10.1145/3510003.3510095.

Linyi Li, Yuhao Zhang, Luyao Ren, Yingfei Xiong, and Tao Xie. Reliability assurance for deep neural network architectures against numerical defects. In *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering*, pp. 1827–1839. IEEE, 2023. URL https://doi.org/10.1109/ICSE48619.2023.00156.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf.

Ling Team, Binwei Zeng, Chao Huang, Chao Zhang, Changxin Tian, Cong Chen, Dingnan Jin, Feng Yu, Feng Zhu, Feng Yuan, Fakang Wang, Gangshan Wang, Guangyao Zhai, Haitao Zhang, Huizhong Li, Jun Zhou, Jia Liu, Junpeng Fang, Junjie Ou, Jun Hu, Ji Luo, Ji Zhang, Jian Liu, Jian Sha, Jianxue Qian, Jiewei Wu, Junping Zhao, Jianguo Li, Jubao Feng, Jingchao Di, Junming Xu, Jinghua Yao, Kuan Xu, Kewei Du, Longfei Li, Lei Liang, Lu Yu, Li Tang, Lin Ju, Peng Xu, Qing Cui, Song Liu, Shicheng Li, Shun Song, Song Yan, Tengwei Cai, Tianyi Chen, Ting Guo, Ting Huang, Tao Feng, Tao Wu, Wei Wu, Xiaolu Zhang, Xueming Yang, Xin Zhao, Xiaobo Hu, Xin Lin, Yao Zhao, Yilong Wang, Yongzhen Guo, Yuanyuan Wang, Yue Yang, Yang Cao,

Yuhao Fu, Yi Xiong, Yanzhe Li, Zhe Li, Zhiqiang Zhang, Ziqi Liu, Zhaoxin Huan, Zujie Wen, Zhenhang Sun, Zhuoxuan Du, and Zhengyu He. Every FLOP counts: Scaling a 300B mixture-of-experts LING LLM without premium GPUs. *arXiv preprint arXiv:2503.05139*, 2025. doi: 10.48550/arXiv.2503.05139. URL https://arxiv.org/abs/2503.05139.

Jiawei Liu, Jinkun Lin, Fabian Ruffy, Cheng Tan, Jinyang Li, Aurojit Panda, and Lingming Zhang. NNSmith: Generating diverse and valid test cases for deep learning compilers. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 530–543. ACM, 2023a. URL https://doi.org/10.1145/3575693.3575707.

Jiawei Liu, Jinjun Peng, Yuyao Wang, and Lingming Zhang. NeuRI: Diversifying DNN generation via inductive rule inference. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 657–669. ACM, 2023b. URL https://doi.org/10.1145/3611643.3616337.

Yunchi Lu, Youshan Miao, Cheng Tan, Peng Huang, Yi Zhu, Xian Zhang, and Fan Yang. TrainVerify: Equivalence-based verification for distributed LLM training. In *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles*, pp. 237–253. ACM, 2025. URL https://doi.org/10.1145/3731569.3764850.

Dolores Miao, Ignacio Laguna, and Cindy Rubio-González. Expression isolation of compiler-induced numerical inconsistencies in heterogeneous code. In *High Performance Computing: ISC High Performance 2023*, volume 13948 of *Lecture Notes in Computer Science*, pp. 381–401. Springer Nature Switzerland, 2023. URL https://doi.org/10.1007/978-3-031-32041-5_20.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.

OpenAI. GPT-6 Astra system card. Official system card, 2026. URL https://deploymentsafety.openai.com/gpt-6-astra. Accessed September 26, 2026.

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

PyTorch Contributors. Numerical accuracy. PyTorch documentation, 2026a. URL https://docs.pytorch.org/docs/2.14/notes/numerical_accuracy.html. Version 2.14; accessed September 16, 2026.

PyTorch Contributors. Optimizing model parameters. PyTorch Tutorials, 2026b. URL https://docs.pytorch.org/tutorials/beginner/basics/optimization_tutorial.html. Accessed September 16, 2026.

Qwen Team. Qwen3.8-Flash-Next. Official Hugging Face model card, 2026. URL https://huggingface.co/Qwen/Qwen3.8-Flash-Next. Accessed September 20, 2026.

Alex Sanchez-Stern, Pavel Panchekha, Sorin Lerner, and Zachary Tatlock. Finding root causes of floating point error. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 256–269. ACM, 2018. URL https://doi.org/10.1145/3192366.3192411.

Dipankar Sarkar. Operator-aware mixed-precision tolerance calibration for tensor kernels. *arXiv preprint arXiv:2607.16228*, 2026. doi: 10.48550/arXiv.2607.16228. URL https://arxiv.org/abs/2607.16228v2. Version 2, revised July 21, 2026.

Shaila Sharmin, Anwar Hossain Zahid, Subhankar Bhattacharjee, Chiamaka Igwilo, Miryung Kim, and Wei Le. Automatically detecting numerical instability in machine learning applications via soft assertions. *Proceedings of the ACM on Software Engineering*, 2(FSE):2806–2827, 2025. URL https://doi.org/10.1145/3729394.

Thang Tran and Lan Dang. Cross-stack validation of language-model training: A clinical fine-tuning case study. *arXiv preprint arXiv:2608.24267*, 2026. doi: 10.48550/arXiv.2608.24267. URL https://arxiv.org/abs/2608.24267.

Gianluigi Vitale. DriftBench: Measuring and predicting infrastructure drift in LLM serving systems. In *Proceedings of Machine Learning and Systems*, volume 8, pp. 1951–1978. MLSys, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/file/ea0b5818ae9255ee1fb1e3b4442d2ffe-Paper-Conference.pdf.

Jiannan Wang, Hung Viet Pham, Qi Li, Lin Tan, Yu Guo, Adnan Aziz, and Erik Meijer. D^3: Differential testing of distributed deep learning with model generation. *IEEE Transactions on Software Engineering*, 51(1):38–52, 2025. URL https://doi.org/10.1109/TSE.2024.3461657.

Zan Wang, Ming Yan, Junjie Chen, Shuang Liu, and Dongdi Zhang. Deep learning library testing via effective model generation. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 788–799. ACM, 2020. URL https://doi.org/10.1145/3368089.3409761.

Peichen Xie, Shuotao Xu, Yang Wang, Fan Yang, and Mao Yang. Bit-accurate modeling of GPU matrix multiply-accumulate units: Demystifying numerical discrepancy and accuracy. *arXiv preprint arXiv:2511.10909*, 2025. doi: 10.48550/arXiv.2511.10909. URL https://arxiv.org/abs/2511.10909v2. Version 2, revised April 14, 2026; first version published in 2025.

Chenyuan Yang, Yinlin Deng, Jiayi Yao, Yuxing Tu, Hanchi Li, and Lingming Zhang. Fuzzing automatic differentiation in deep-learning libraries. In *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering*, pp. 1174–1186. IEEE, 2023. URL https://doi.org/10.1109/ICSE48619.2023.00105.

Anwar Hossain Zahid, Ignacio Laguna, and Wei Le. Testing GPU numerics: Finding numerical differences between NVIDIA and AMD GPUs. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 547–557. IEEE, 2024. doi: 10.1109/SCW63240.2024.00077. URL https://sc24.supercomputing.org/proceedings/workshops/workshop_pages/ws_hpct106.html.

Andreas Zeller and Ralf Hildebrandt. Simplifying and isolating failure-inducing input. *IEEE Transactions on Software Engineering*, 28(2):183–200, 2002. doi: 10.1109/32.988498. URL https://www.st.cs.uni-saarland.de/papers/tse2002/.

Qi Zhan, Xing Hu, Yuanyi Lin, Tongtong Xu, Xin Xia, and Shanping Li. When AllClose fails: Round-off error estimation for deep learning programs. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 91–103. IEEE, 2025. doi: 10.1109/ASE63991.2025.00016. URL https://qi-zhan.github.io/assets/pdf/ASE2025zq.pdf.

Wenxin Zheng, Wenxiao Wang, Yun Zhang, Mingcong Han, Bin Xu, Jinyu Gu, Xingda Wei, Haibo Chen, Zuquan Song, Gaohong Liu, Yucheng Nie, Zhe Nan, Zhuolin Zheng, Huan Yu, Shuguang Wang, Ziming Zhou, Hang Zhu, Wencong Xiao, and Xin Liu. SDCs in the wild: Characterizing and diagnosing SDC-Defective GPUs in production LLM training (operational systems). In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, pp. 1349–1367. USENIX Association, 2026. URL https://www.usenix.org/conference/osdi26/presentation/zheng.

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. URL https://www.usenix.org/conference/osdi26/presentation/zhou-ziming.

A Diagnostic Metrics Definitions

This appendix provides the mathematical definitions for the diagnostic metrics introduced in Section 4.1.

A.1 Numerical Value Consistency

For finite matched tensors x and y with n elements, exact and tolerance-based checks are

$$E(x,y) = \mathbf{1}[\forall i,\ x_i = y_i]\,, \quad A(x,y) = \mathbf{1}[\forall i,\ |x_i - y_i| \le a + r|y_i|]\,, \tag{A.1}$$

where a and r are absolute and relative tolerances. Deviation measures quantify the largest local and overall differences:

$$D_{\max}(x,y) = \max_i |x_i - y_i|, \quad D_{\mathrm{rel}}(x,y) = \frac{\|y-x\|_2}{\|x\|_2}. \tag{A.2}$$

Special-value counts record infinities:

$$C_\infty(y) = \sum_{i=1}^{n} \mathbf{1}[y_i \in \{-\infty, +\infty\}]\,. \tag{A.3}$$

A.2 Optimizer State Consistency

For Adam with gradient g_t and pre-update states m_{t-1}, v_{t-1}, expected states are

$$m_t^\star = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad v_t^\star = \beta_2 v_{t-1} + (1-\beta_2) g_t^2. \tag{A.4}$$

Residuals measure deviations from expected evolution:

$$R_m = \frac{\|m_t - m_t^\star\|_2}{\max(\|m_t^\star\|_2, \epsilon_d)}, \quad R_v = \frac{\|v_t - v_t^\star\|_2}{\max(\|v_t^\star\|_2, \epsilon_d)}. \tag{A.5}$$

A.3 Update Behavior Consistency

For AdamW, the expected parameter update is

$$\theta_t^\star = (1-\eta\lambda)\theta_{t-1} - \eta \frac{m_t^\star/(1-\beta_1^t)}{\sqrt{v_t^\star/(1-\beta_2^t)} + \varepsilon_{\mathrm{opt}}}. \tag{A.6}$$

Sign disagreement and lost updates assess actual versus expected increments $\Delta\theta_t = \theta_t - \theta_{t-1}$:

$$B_{\mathrm{sign}} = \frac{1}{|S|} \sum_{i \in S} \mathbf{1}\big[\mathrm{sgn}(\Delta\theta_{t,i}) \neq \mathrm{sgn}(\Delta\theta_{t,i}^\star)\big]\,, \quad B_{\mathrm{lost}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\big[\Delta\theta_{t,i} = 0 \ \wedge\ \Delta\theta_{t,i}^\star \neq 0\big]\,, \tag{A.7}$$

where $S = \{i : \Delta\theta_{t,i}^\star \neq 0\}$.

B Detailed Alignment Rules and Decision Procedures

This appendix specifies the calibration and runtime rules used in Section 4.2.

B.1 Assessment scope and rule configuration

An execution pair is the unit of diagnosis; a semantic-boundary observation supplies evidence, and a lineage group is the unit of data splitting. Observations with shared origins remain grouped during calibration and validation.

For execution pair p, the frozen configuration specifies required checks $\mathcal{R}_p$, optional checks, semantic mappings, applicable roles, metrics, thresholds, and composition rules. These are fixed before observing test outcomes.

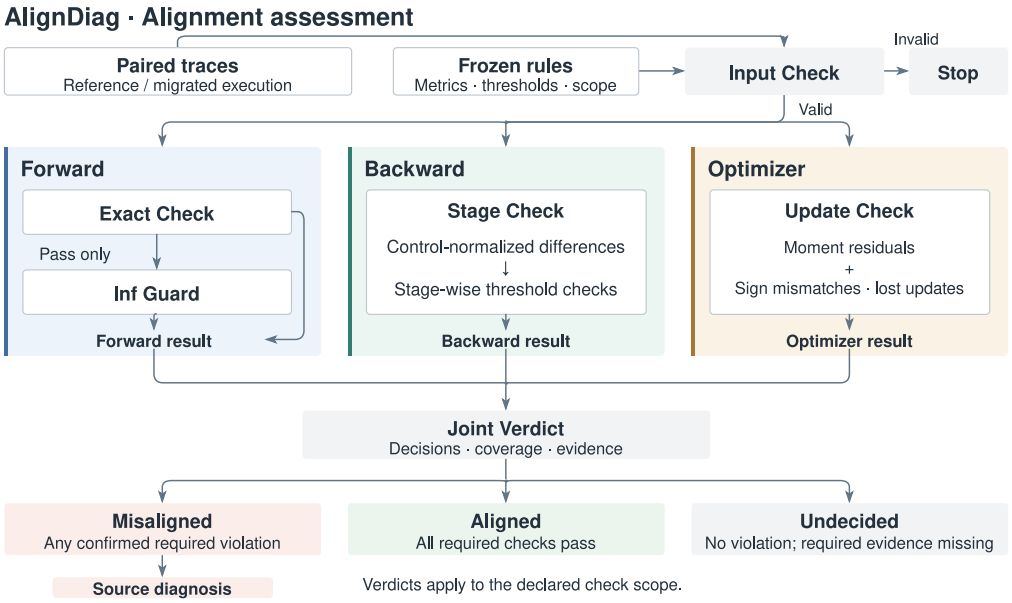

Figure B.1: **Alignment assessment.** The three modules return decisions, coverage, and evidence; the joint verdict combines their assessments.

The forward, backward, and optimizer modules return decisions $d_m \in \{\mathrm{A}, \mathrm{M}, \mathrm{U}\}$ (aligned, misaligned, undecided), completed checks, and supporting observations (Figure B.1). Inapplicable checks retain their reasons and do not satisfy required coverage.

For a required check r, let $c_r = 1$ indicate that it has been completed with valid evidence. The fraction

$$C_p = \frac{1}{|\mathcal{R}_p|} \sum_{r \in \mathcal{R}_p} c_r \tag{B.1}$$

is check-level coverage, distinct from pair- or group-level decision coverage. Alignment requires non-empty required scopes.

B.2 Metric selection and threshold calibration

Label–metric dependency audits separate labeling evidence from evaluated metrics. For candidate metric v_i, scores are oriented so larger values indicate disagreement: $z_i = sv_i$, where $s \in \{+1, -1\}$. A violation occurs at $z_i > t$.

Forward calibration selects thresholds by maximizing group-balanced recall:

$$t^\star = \arg\max_{t \in \mathcal{T}} \left(\frac{R_{\mathrm{A}}(t) + R_{\mathrm{M}}(t)}{2}, \min\{R_{\mathrm{A}}(t), R_{\mathrm{M}}(t)\}, -|t| \right), \tag{B.2}$$

where $R_{\mathrm{A}}(t)$ and $R_{\mathrm{M}}(t)$ are weighted recalls for aligned and misaligned classes. Weaker-class recall and boundary magnitude break ties. Selected rules are validated independently and frozen before testing.

Backward and optimizer calibration use empirical envelopes from permitted-variation records:

$$\tau_{\mathrm{B}} = \max_{p \in \mathcal{D}_{\mathrm{cal,A}}} \{\text{max registered scores of } p\}, \quad \tau_{\mathrm{O}} = \max\{10^{-7}, 4r_{\max}\}, \tag{B.3}$$

where $r_{\max}$ pools optimizer-state residuals over the calibration set.

B.3 Forward assessment

Exact Check and Inf Guard use separate registered sets (Figures B.2–B.3).

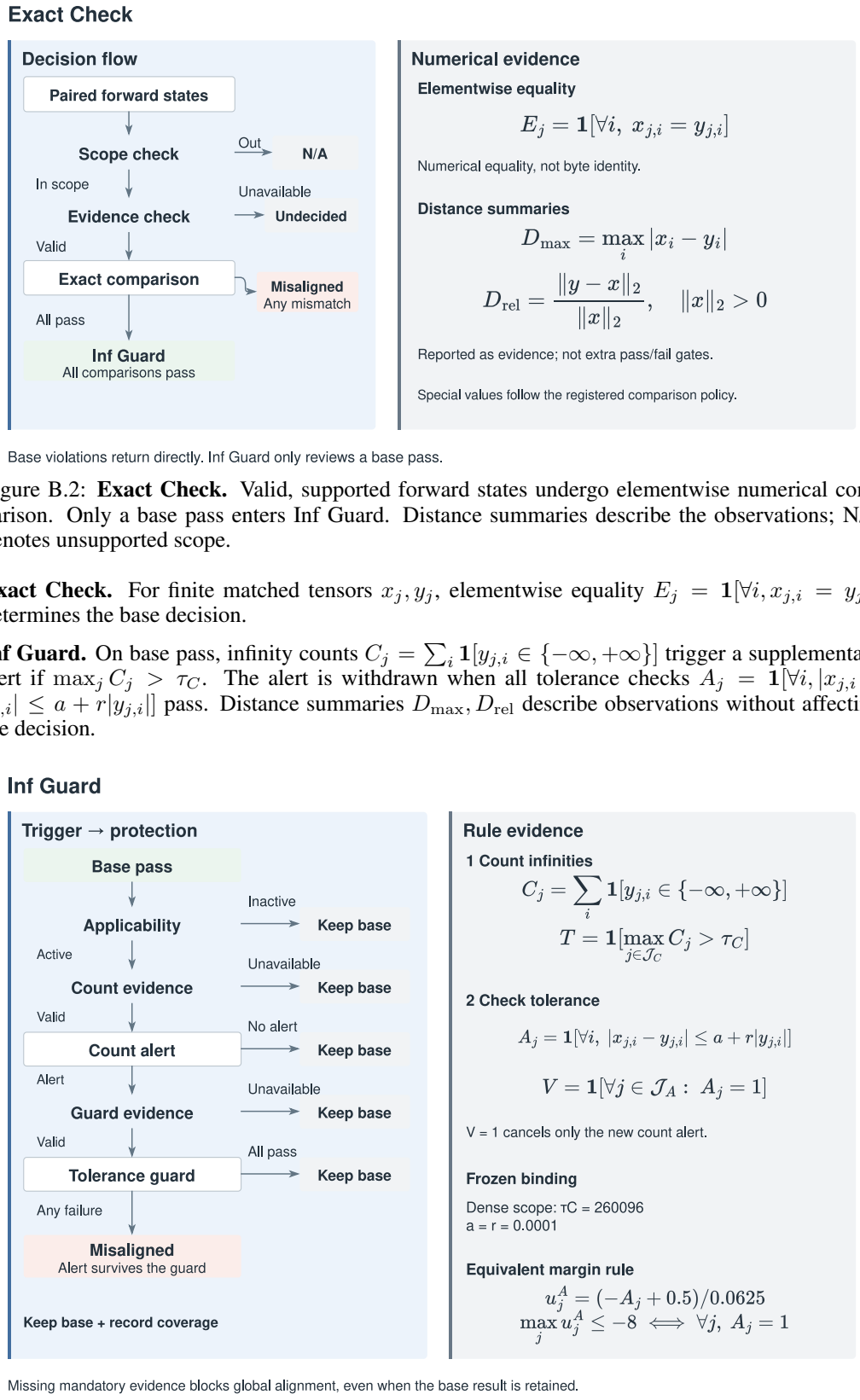

Figure B.2: **Exact Check.** Valid, supported forward states undergo elementwise numerical comparison. Only a base pass enters Inf Guard. Distance summaries describe the observations; N/A denotes unsupported scope.

Exact Check. For finite matched tensors x_j, y_j, elementwise equality $E_j = \mathbf{1}[\forall i, x_{j,i} = y_{j,i}]$ determines the base decision.

Inf Guard. On base pass, infinity counts $C_j = \sum_i \mathbf{1}[y_{j,i} \in \{-\infty, +\infty\}]$ trigger a supplementary alert if $\max_j C_j > \tau_C$. The alert is withdrawn when all tolerance checks $A_j = \mathbf{1}[\forall i, |x_{j,i} - y_{j,i}| \le a + r|y_{j,i}|]$ pass. Distance summaries $D_{\max}, D_{\mathrm{rel}}$ describe observations without affecting the decision.

Figure B.3: **Inf Guard.** Infinity counts trigger an additional alert, which the tolerance guard can cancel. Missing supplementary evidence or cancelled alerts do not override base violations.

B.4 BACKWARD STAGE-CHAIN ASSESSMENT

Backward checks compare gradient stages against matched controls (Figure B.4). For target stage j with control pool $\mathcal{C}_j$, the deviation score is

$$s_j = \log_{10} \frac{D_{\max}(x_j, y_j) + \epsilon_s}{\max_{c \in \mathcal{C}_j} D_{\max}(x_j, c_j) + \epsilon_s}. \tag{B.4}$$

The module examines the current gradient and its registered upstream stages jointly, reporting misalignment if any valid score exceeds τ_{B}.

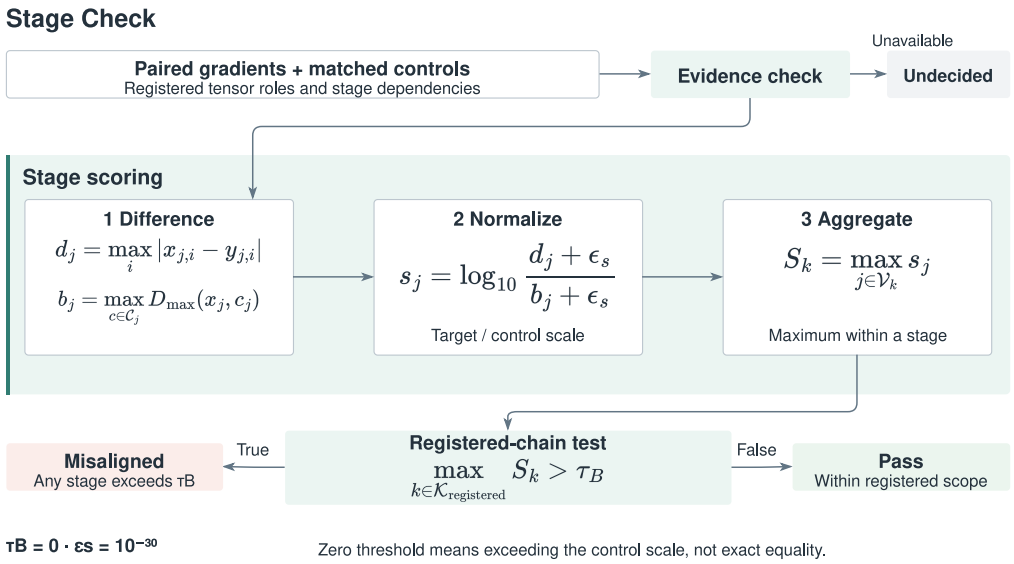

Figure B.4: **Stage Check.** Control-normalized differences are aggregated within stages and tested along registered dependencies. A zero score boundary means exceeding the matched-control scale, not requiring exact equality.

B.5 OPTIMIZER-STATE AND UPDATE ASSESSMENT

Update Check (Figure B.5) validates optimizer states and parameter updates against declared equations.

Given gradient g_t and pre-update states m_{t-1}, v_{t-1}, expected states follow the declared equations:

$$m_t^\star = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad v_t^\star = \beta_2 v_{t-1} + (1-\beta_2) g_t^2. \tag{B.5}$$

Residuals R_m, R_v measure deviations from expected evolution. The expected parameter update is

$$\theta_t^\star = (1 - \eta\lambda)\theta_{t-1} - \eta \frac{m_t^\star/(1-\beta_1^t)}{\sqrt{v_t^\star/(1-\beta_2^t)} + \varepsilon_{\text{opt}}}. \tag{B.6}$$

Sign disagreement B_{sign} and lost updates B_{lost} assess the actual parameter increment $\Delta\theta_t = \theta_t - \theta_{t-1}$ against the expected $\Delta\theta_t^\star$. Any threshold violation among the four checks establishes misalignment.

B.6 JOINT VERDICT AND DIAGNOSTIC HANDOFF

For required check r, let $v_r = 1$ denote a confirmed violation after module-level validation and protection, $a_r = 1$ a completed pass, and $c_r = 1$ valid coverage. The joint verdict is

$$D_p = \begin{cases} \mathrm{M}, & \exists r \in \mathcal{R}_p : v_r = 1, \\ \mathrm{A}, & |\mathcal{R}_p| > 0 \ \wedge \ \forall r \in \mathcal{R}_p : c_r = 1 \wedge a_r = 1, \\ \mathrm{U}, & \text{otherwise.} \end{cases} \tag{B.7}$$

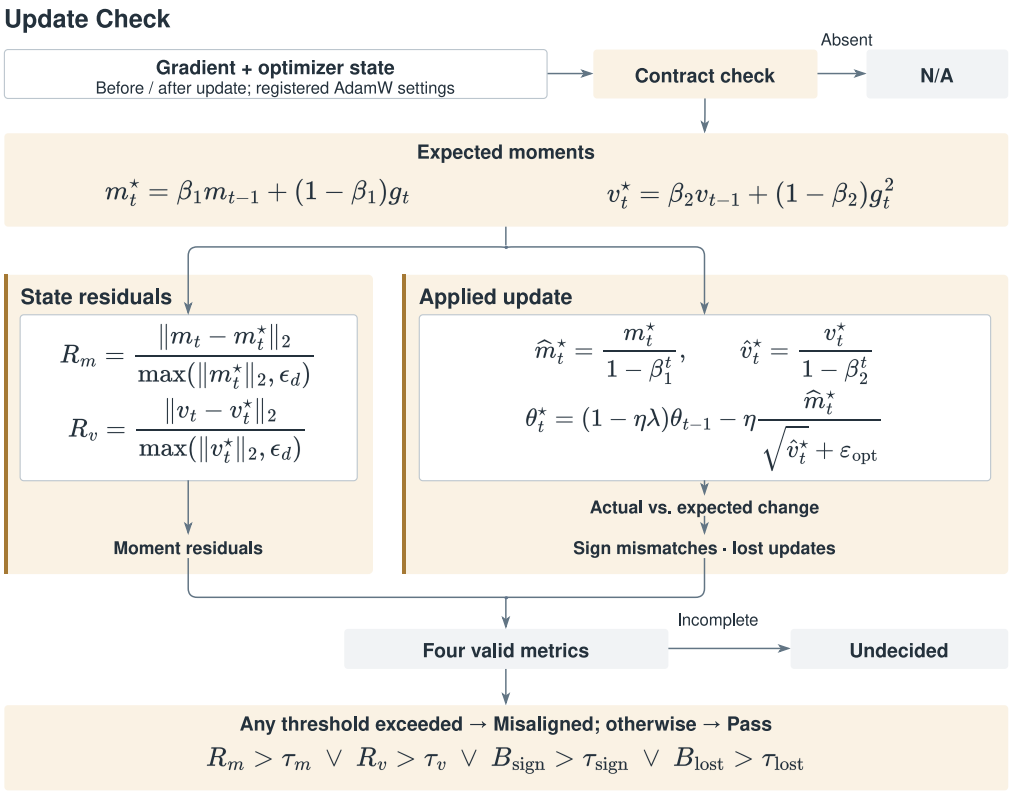

Figure B.5: **Update Check.** Shared expected moments support state-residual and parameter-update checks. Four complete, valid metrics feed the joint threshold rule; sign mismatches and lost updates are related evidence, not independent votes.

Confirmed violations take priority over passes or missing evidence elsewhere. Cancelled supplementary alerts are excluded.

The diagnostic handoff records pair and rule identities, module decisions, coverage, triggering states, metric values and thresholds, dependencies, and alert cancellations. It supplies observations for source localization.

B.7 Reproducible configuration

The record binds thresholds to metric producers, state scopes, mappings, splits, and missing-value policies. Forward base/count/guard boundaries are $(-0.5, 260096, -0.5)$, with scales $(1, 32512, 0.0625)$; margins are clipped to $[-16, 16]$, and the guard cutoff is -8. The backward boundary is 0; optimizer hyperparameters come from the assessed execution.

The versioned record must specify NaN, infinity, signed-zero, dtype-conversion, and computation-precision semantics, together with hashes and control-selection keys. Scope is rechecked after changes in model shape, state semantics, numerical format, or backend.

Regression tests cover equality at strict thresholds, empty/invalid evidence, duplicate optimizer fields, alert cancellation, and missing required checks. Missing evidence plus a confirmed violation yields misalignment; missing evidence alone yields undecided.

C Multi-Source Localization and Intervention Validation

This appendix specifies the multi-source localization procedure. Figure C.1 shows the workflow: local checks identify candidates through cumulative neutralization, followed by recovery and attribution assessments.

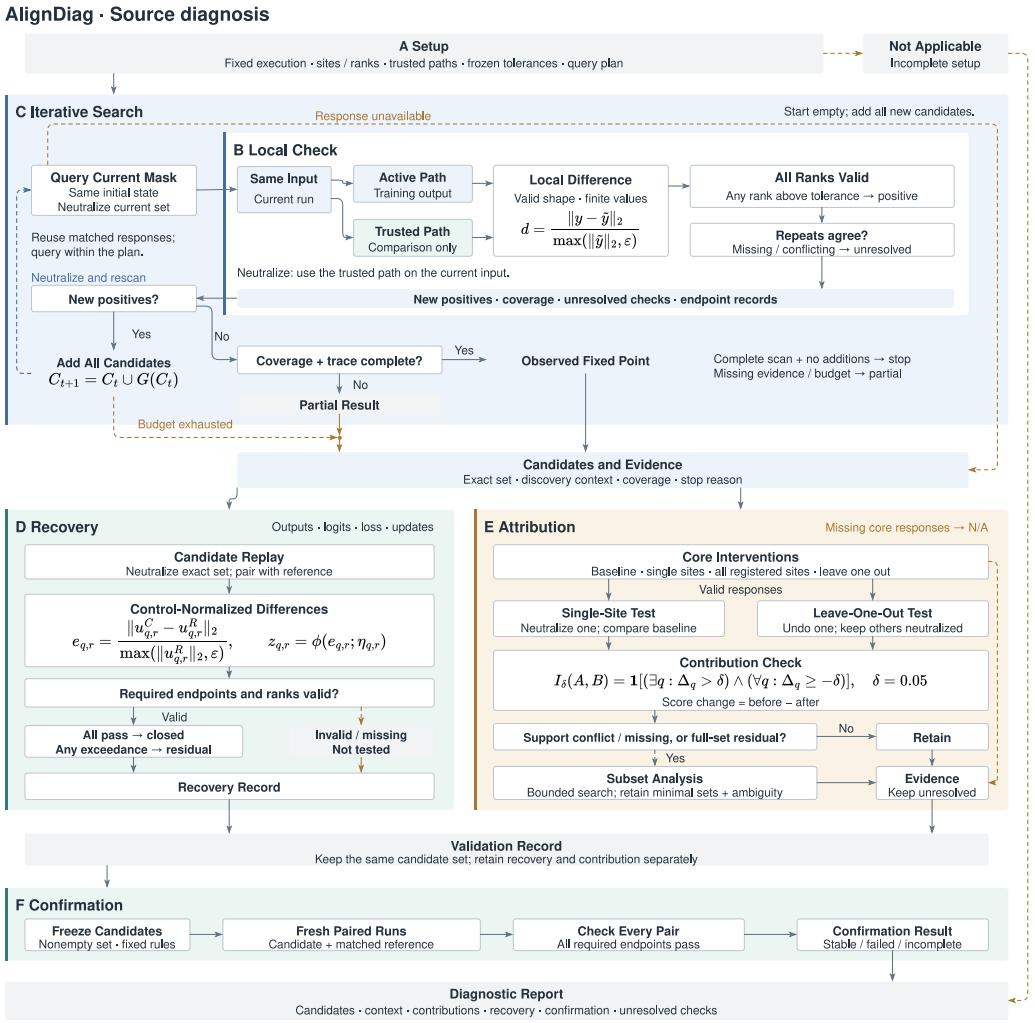

Figure C.1: **Localization and validation.** Local Check supplies each search round with new candidates and current coverage. Cumulative neutralization exposes residual sources; Recovery and Attribution assess complementary intervention effects before fresh paired Confirmation.

C.1 Execution Setup

Each execution group fixes model, checkpoint, inputs, states, and topology. Registered sites $V = \{s_1, \ldots, s_m\}$ are matched by step, layer, and role. A mask $M \subseteq V$ selects sites to neutralize by replacing active implementations with trusted references.

C.2 SAME-INPUT CHECKS

At each site s, both active and trusted paths receive the same input, producing outputs y and $\widetilde{y}$. The relative distance is

$$d_{s,r}^{M,j} = \frac{\|\text{diff}\|_2}{\max(\|\widetilde{y}\|_2, \varepsilon)}. \tag{C.1}$$

Site s is flagged when $d_{s,r}^{M,j} > \tau_s$ for any rank, with tolerance τ_s calibrated from control runs. The query returns sites with stable decisions:

$$G(M) = \{s \in V \setminus M : B_s(M) = 1\}. \tag{C.2}$$

C.3 ITERATIVE SEARCH

Starting from $C_0 = \varnothing$, each round adds new positives: $C_{t+1} = C_t \cup G(C_t)$. The search stops when no new positives appear. At most m productive rounds visit $m + 1$ masks.

C.4 ENDPOINT RECOVERY

For required endpoints Q, recovery compares mask M outputs with matched references:

$$e_{q,r}^{M,j} = \frac{\|u_{q,r}^{M,j} - u_{q,r}^{R,j}\|_2}{\max(\|u_{q,r}^{R,j}\|_2, \varepsilon)}. \tag{C.3}$$

Recovery requires every endpoint to remain within calibrated tolerance $\eta_{q,r}$:

$$\kappa(M) = \begin{cases} \text{closed}, & \forall q \in Q, r \in R : e_{q,r}^{M,j} \le \eta_{q,r}, \\ \text{residual}, & \text{otherwise}. \end{cases} \tag{C.4}$$

C.5 CONTRIBUTION ANALYSIS

Contribution compares endpoint deviations between interventions. For compatible endpoint vectors, $\Delta_q(A, B) = \overline{z}_q(A) - \overline{z}_q(B)$ measures improvement when neutralizing B versus A. Single-site and leave-one-out tests identify candidates with measurable contributions.

C.6 FRESH CONFIRMATION

Fresh paired runs validate the candidate set with new executions. Each pair's recovery decision is

$$r_j(C) = \mathbf{1}[\forall (q, r) : e_{q,r}^{C,j} \le \eta_{q,r}]. \tag{C.5}$$

The protocol schedules three pairs; all must pass for confirmation.

C.7 EXECUTION RECORDS AND EVALUATION

Source correctness is evaluated after predictions are frozen:

$$\text{Precision} = \frac{|P \cap T|}{|P|}, \quad \text{Recall} = \frac{|P \cap T|}{|T|}, \quad \text{ExactSet} = \mathbf{1}[P = T]. \tag{C.6}$$

Reports link candidates to their discovery context, contribution evidence, and confirmation outcomes.

D DIAGNOSTIC EVALUATION: SETTINGS AND ADDITIONAL RESULTS

D.1 MODEL-FAMILY DETECTION RECALL

D.2 CONTROLLED FAULT CONSTRUCTION AND LOCALIZATION SETTINGS

We run Qwen3-0.6B on H200 and Ascend NPU with four ranks (DP=2, CP=2), two training steps, sequence length 512, and global batch size 16. The registered sites are attention RMSNorm in layers

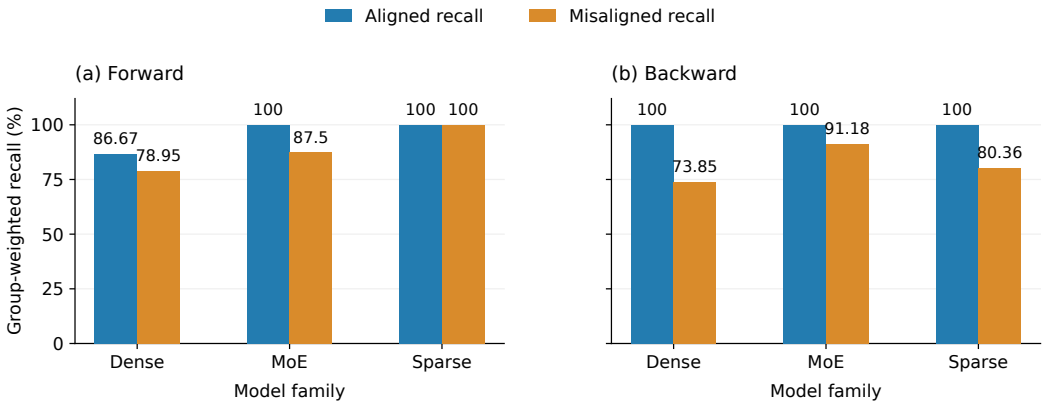

Figure D.1: Detection recall by model family. Aligned and misaligned recalls are averaged equally over experimental groups containing the respective class. Forward uses the forward evaluation cohort; Backward uses the complete backward stage-chain evaluation. Dense, MoE, and Sparse contain 19/18/16 groups (366/464/28 pairs) for Forward and 10/6/8 groups (418/424/272 pairs) for Backward, respectively.

Table 3: Controlled localization settings. Layer indices are zero-based.

Item	Setting
Model and platforms	Qwen3-0.6B; paired NVIDIA H200 and Ascend NPU runs
Training window	Two training steps; four ranks; DP=2, CP=2
Inputs and initialization	Sequence length 512; global batch size 16; seed 42; two fixed token batches and a fixed initial training state
Registered sites	Attention RMSNorm in layers 0 and 27; rotary position embedding and Softmax in layer 14
Fault construction	Controlled precision changes violating the declared FP32 computation requirement at the selected sites
Test inputs	32 independent inputs: 24 faulty and eight clean; the same paired inputs on both platforms
Subset coverage	All 16 site subsets; 8/6/10/7/1 inputs with 0/1/2/3/4 sources
Calibration and validation	Eight calibration groups and eight validation groups, excluded from the test inputs
Scan coverage	Four sites $\times$ four ranks $\times$ two steps: 32 comparison units per scan
Decision and stopping	Same-input local relative-L2 comparison; threshold fixed at zero after calibration; cumulative neutralization; stop after a scan with no new source
Scoring	Compare the frozen predicted set with injection ground truth; injected locations and counts are not supplied to the search

0 and 27, rotary position embedding in layer 14, and Softmax in layer 14, using zero-based indices. Controlled precision changes violate the FP32 computation requirement declared for these sites. The 32 test inputs cover all 16 site subsets, with 8, 6, 10, 7, and 1 inputs containing zero through four sources, respectively. Eight calibration and eight validation groups are separate from the test inputs. Each scan checks all four sites across all ranks and steps using the current input for both active and trusted computation. The calibrated local relative-L2 threshold is zero in this controlled setting. Detected sources are cumulatively neutralized, and a complete scan with no new source establishes termination. The recorded source set is scored against injection ground truth.

D.3 Offline Analysis Cost and Evidence Reuse

We measure offline analysis cost using the complete registered workloads, since total time depends on both evidence volume and processing requirements (Appendix Figures D.3 and D.4). Full refitting and evaluation take 98.35 s for forward assessment and 99.14 s for backward assessment. Once the rules are fixed, evaluation over all 858 forward and 1,114 backward pairs takes a median of 51.00 s and 18.67 s, respectively. Repeated use avoids paying the rule-learning and selection cost again. The larger backward cohort takes less time to evaluate.

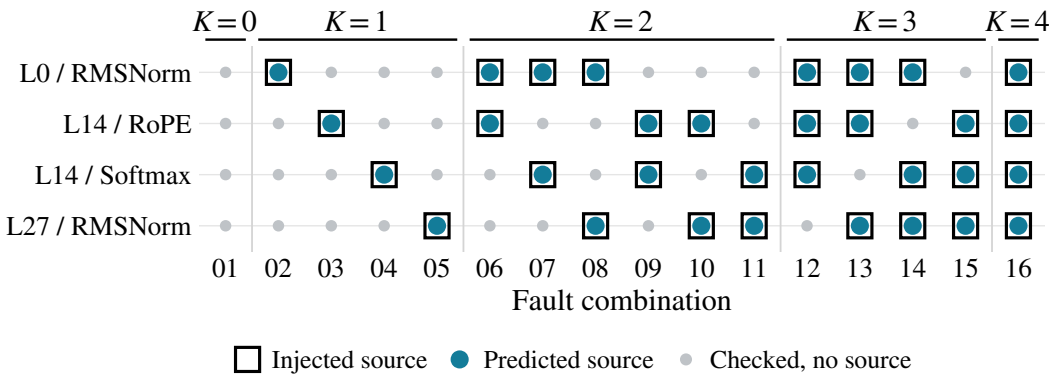

Figure D.2: Controlled source localization in Qwen3-0.6B. The test covers all 16 subsets of four registered operator sites using 32 paired inputs on H200 and NPU. On each platform, all 24 faulty inputs yield the exact injected source set, covering 51 source occurrences without extra predictions; all eight clean inputs produce no source. Faulty inputs require one discovery scan and one stopping-confirmation scan.

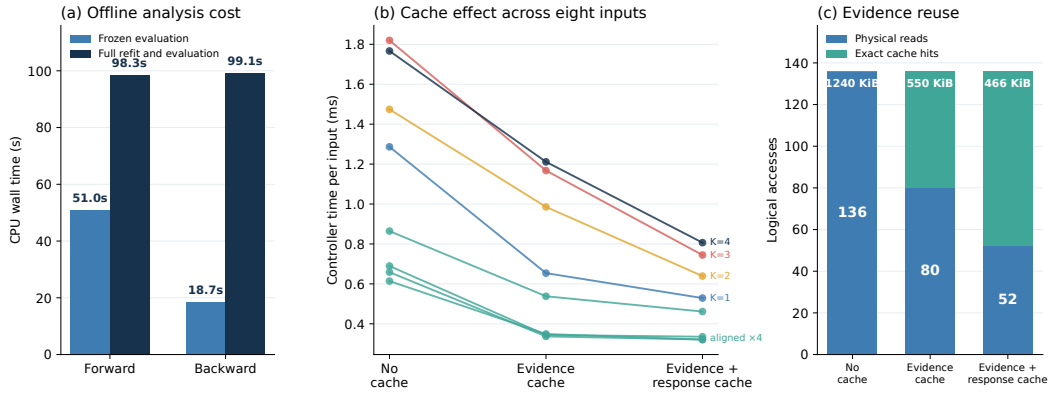

Figure D.3: Offline analysis cost and evidence reuse. (a) Full refitting and evaluation versus fixed-rule evaluation on the complete registered forward and backward workloads; fixed-rule bars report medians over ten independent processes. (b) Controller time for eight independent inputs, using paired-hardware means within each repetition and medians across ten repetitions. Four aligned inputs stop after detection; faulty inputs contain $K = 1$ to 4 sources. (c) The same 136 logical accesses are served by physical reads or exact cache hits. Total controller replay takes 18.82, 12.11, and 8.81 ms with no cache, evidence caching, and evidence plus response caching, respectively. All modes preserve diagnostic outputs. Times measure CPU analysis of saved evidence, excluding device execution and evidence generation.

On saved evidence for eight diagnostic cases with paired H200/NPU records, controller replay takes 18.82 ms without application-level caching. Caching observations reduces this to 12.11 ms; additionally reusing responses to identical requests reduces it to 8.81 ms. Physical reads fall from 136 to 52 while the diagnostic outputs remain unchanged. Observation and response reuse thus remove complementary sources of repeated work, providing a 2.14× controller speedup over the same diagnostic workload.

Controller timing uses paired-hardware means within each repetition and medians across ten repetitions. The reported 18.82/12.11/8.81 ms values measure replay of the complete eight-case workload. K denotes the number of sources and is used only for result stratification. Timing measures CPU analysis of saved evidence, excluding GPU/NPU training and physical interventions. Stage medians need not sum to the median total time.

D.4 Repair Plot Definitions and Cohorts

Figure 2. Repair effects in a three-source Qwen3-0.6B case. (a) Repair subsets on H200; each node reports baseline-normalized logit, loss, and parameter-update errors, in that order. RMSNorm is in

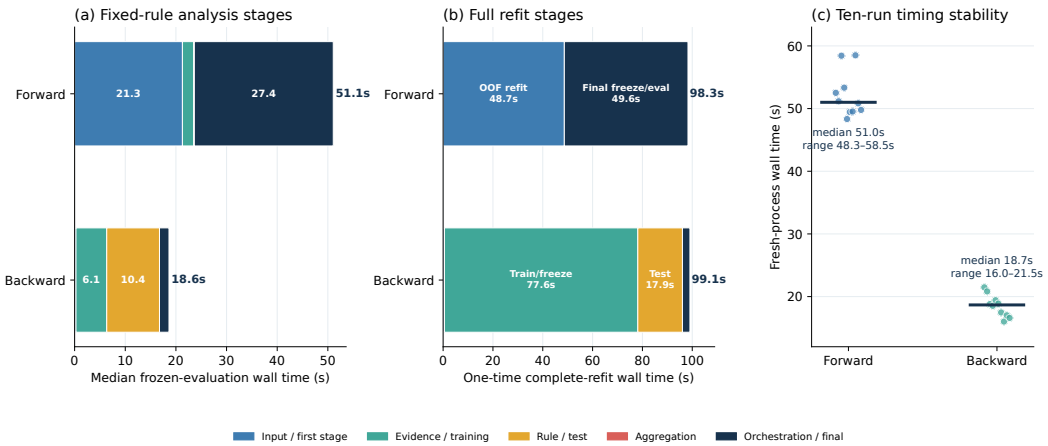

Figure D.4: Offline cost decomposition and timing stability. (a) Median times for stages of fixed-rule analysis. Sums of stage medians can differ from the median total time. (b) Stage times for a complete refit, including out-of-fold (OOF) calibration or training, rule freezing, and evaluation. (c) Fixed-rule evaluation times from ten independent processes, with median and observed range. The operating-system page cache is retained. Repetitions measure timing variability on the same workloads.

layer 0; RoPE and Softmax are in layer 14. (b) Loss responses on H200 and NPU. Repairing RMSNorm alone increases NPU loss error to 3.11× baseline, whereas repairing all three sites restores every checked endpoint on both platforms. Each subset is executed independently from the same initial state. Edges add one repaired site; curves connect repair conditions. Errors are maximum absolute differences over ranks, steps, and endpoint coordinates, normalized by the corresponding no-repair error; updates cover all 311 trainable parameters.

Figure D.5. Residual errors in three diagnostic endpoints—logits, loss, and parameter updates—under partial and complete repair across 21 completed multi-source hardware–case results (nine H200 and twelve NPU). Columns report logits, loss, and parameter updates. Repair one replaces a single predicted source; Repair all except one withdraws one repair from the complete set; Repair all replaces the full set. Points show measured conditions and lines connect within-case category medians. Blue circles and orange triangles distinguish corrected replications and newly evaluated cases. Errors are normalized by the same-case, same-hardware no-repair baseline; the vertical axis is linear near zero and logarithmic above 0.1. True zeros are retained. For two-source cases, the two partial-repair categories reuse the same measured subsets. These cases belong to the completed 56-of-72 intervention snapshot.

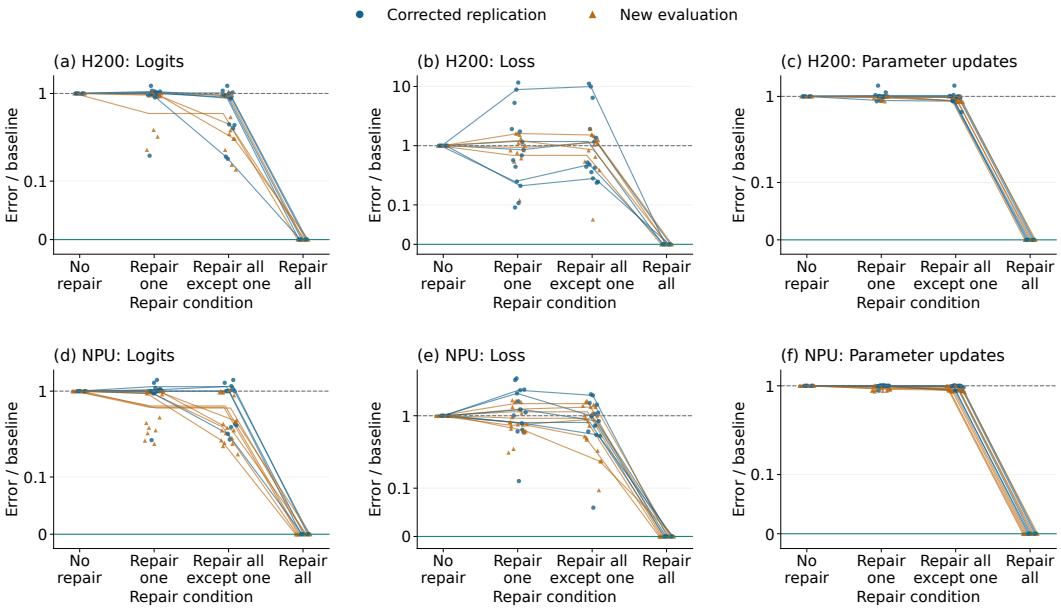

Figure D.5: **Residual errors across 21 multi-source hardware–case results.** Columns show logits, loss, and parameter updates relative to the same-case no-repair baseline. Blue circles and orange triangles distinguish corrected replications and new cases; lines connect within-case category medians. The axis is linear near zero and logarithmic above 0.1; true zeros are retained. Appendix D.4 defines repair subsets and cohort scope.

D.5 Comparison Plot Protocols

Figure 3. Comparison with Prior Numerical Criteria and Calibration Ablations. (a) Forward and (b) backward assessment. Red and blue points denote group-weighted false-positive and fault-recall rates, respectively; horizontal bars show 95% confidence intervals. * denotes one threshold shared across categories; @ denotes thresholds calibrated separately by tensor role in forward assessment and by computation stage in backward assessment. Thresholds are fitted on training data and frozen for evaluation. The figure combines comparisons against representative prior numerical criteria with ablations of metric combination and calibration granularity; the two aspects are discussed in the comparison and ablation subsections, respectively.

Figure D.6. Localization Completeness and Scan Cost. One-at-a-time and All-new both recover the complete source set and finish with a no-new-source scan on NPU. (a) Mean scans for 24 all-visible faulty cases and eight partly masked cases, in which an upstream fault initially hides a downstream source. (b) Per-case scan savings for all-visible cases. (c) Scan cost by source count; the one- through four-source strata contain 6, 10, 7, and 1 cases. (d) Scan budgets needed to complete the target fraction of all-visible cases. All-new reduces scans from 75 to 48 in the all-visible cohort; both strategies require three scans in the partly masked cohort. Initial and final confirmation scans are included.

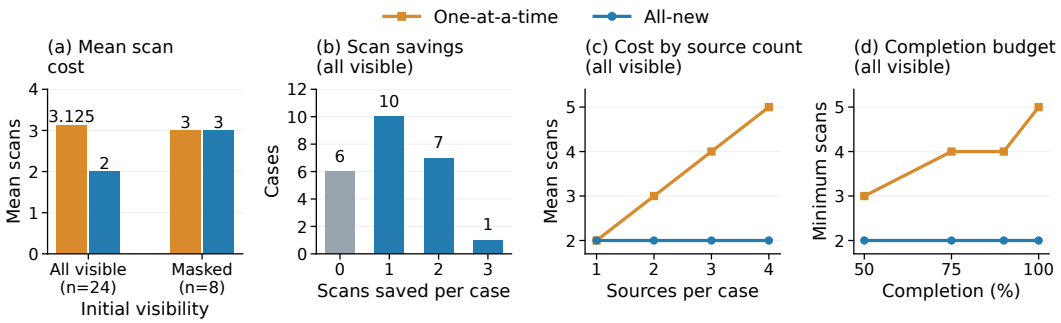

Figure D.6: **Localization completeness and scan cost on NPU.** (a) All-visible and partly masked cases; (b) per-case savings; (c) cost by source count; (d) completion budgets. Both strategies recover all sources and include a final no-new-source scan. All-new reduces all-visible scans from 75 to 48; both require three scans for masked cases. Cohort details appear in Appendix D.5.

D.6 Cross-configuration Detection Results

Table 4: **Cross-configuration detection performance.** (a) Added models: Llama-3.2-3B, OLMo-3-1025-7B, and Granite-4.0-h-tiny-base, retaining the selected metrics and recalibrating numerical parameters before evaluation. (b) Target hardware: NVIDIA RTX 4090 with H200 references after mixed-source adaptation. All values are percentages.

Assessment stage	Recall (%)		Balanced accuracy (%)
	Aligned	Misaligned	
(a) Added models			
Forward	100.00	95.73	97.86
Backward	100.00	89.29	94.64
Overall	**100.00**	**90.90**	**95.45**
(b) Target hardware (RTX 4090)			
Forward	95.56	100.00	97.78
Backward	92.44	93.33	92.89
Overall	**96.00**	**97.33**	**96.67**

E Additional Ablation and Calibration Analysis

E.1 Update-stage Checks

The update-stage analysis compares the complete AlignDiag assessment with individual optimizer checks and their combined rule on 318 reference–target pairs from 22 groups, containing 159 normal and 159 faulty pairs (Appendix Figure E.1). The complete assessment detects all 159 faults without false positives. On the registered cohort, first-moment, second-moment, update-equation, sign, and lost-update checks achieve group-weighted fault recalls of 0.00%, 4.55%, 13.64%, 9.09%, and 4.55%, respectively, all with zero observed false positives. The combined optimizer rule achieves 13.64% group-weighted recall, detecting nine fault pairs. This contrast reflects the scope of the checks, as local optimizer consistency tests whether the update computation respects its given inputs, whereas the complete assessment also uses current-stage and upstream evidence to detect errors propagated through the training chain. An optimizer can satisfy its local equations while consuming incorrect gradients. Local checks provide complementary evidence to stage-chain assessment.

Optimizer evidence is available for 246 of the 318 pairs; unavailable checks remain undecided and are retained in the registered denominators. Consequently, zero observed false positives does not imply a correct normal verdict for every pair. Group-weighted check coverage is 77.27%. The combined optimizer rule uses the first- and second-moment, sign, and lost-update checks. The update-equation residual is evaluated separately, with its threshold set from training-normal residuals before application to the test cohort. Horizontal intervals are 95% group-bootstrap intervals from 10,000 resamples.

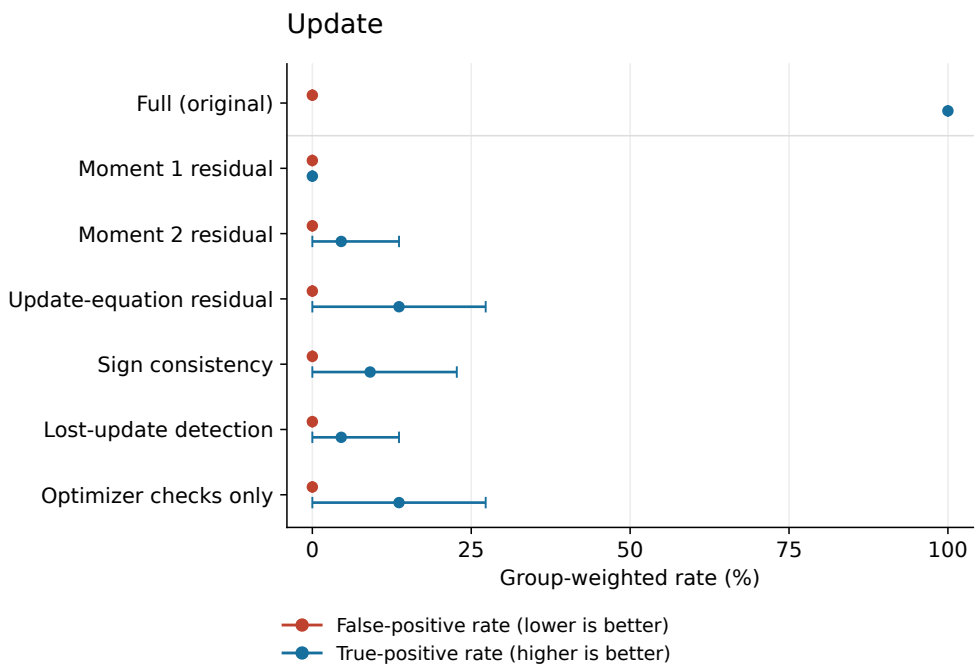

Figure E.1: **Update-stage checks and complete stage-chain assessment.** Full (original) denotes AlignDiag's complete update-stage verdict, including current-stage and upstream evidence. Other rows evaluate individual local checks; Optimizer checks only combines first-moment, second-moment, sign, and lost-update checks, excluding the separately evaluated update-equation residual. Red and blue points denote group-weighted false-positive and fault-recall rates, with 95% group-bootstrap intervals. The cohort contains 22 groups and 318 pairs; local optimizer evidence is available for 246 pairs. Unavailable checks remain in the denominators, and their 77.27% group-weighted coverage is reported here rather than plotted.

E.2 Random Support-set Calibration

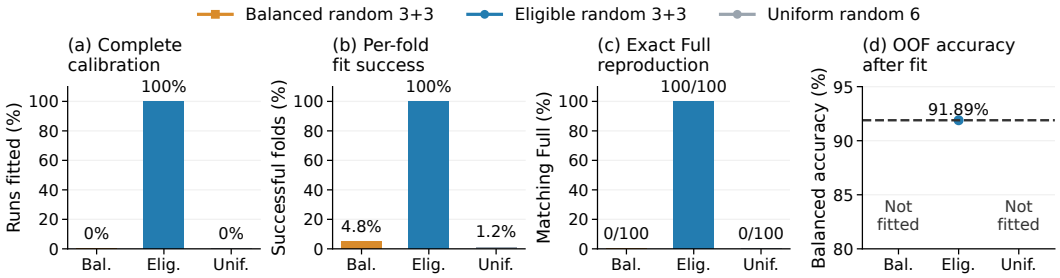

Figure E.2: **Six-shot calibration across 100 random support draws.** (a) Complete five-fold calibration; (b) individual fold fits; (c) full-data recovery; (d) accuracy conditional on successful fitting. Bal., Elig., and Unif. denote the three sampling schemes in the legend. Not fitted denotes unavailable fits, not zero accuracy. Eligibility depends on evidence availability. Details appear in Appendix E.4.

E.3 Component Contributions under Frozen Configurations

E.4 Ablation and Calibration Plot Protocols

Figure E.3. Component contributions under frozen decision configurations. (a,d) Forward assessment on 44 development groups and 840 pairs, replaying the saved rule for each held-out fold. (b,e) Complete backward stage-chain assessment on 24 groups and 1,114 pairs. (c) Update assessment on 59 development groups and 798 pairs. (f) Local optimizer checks on 632 valid records

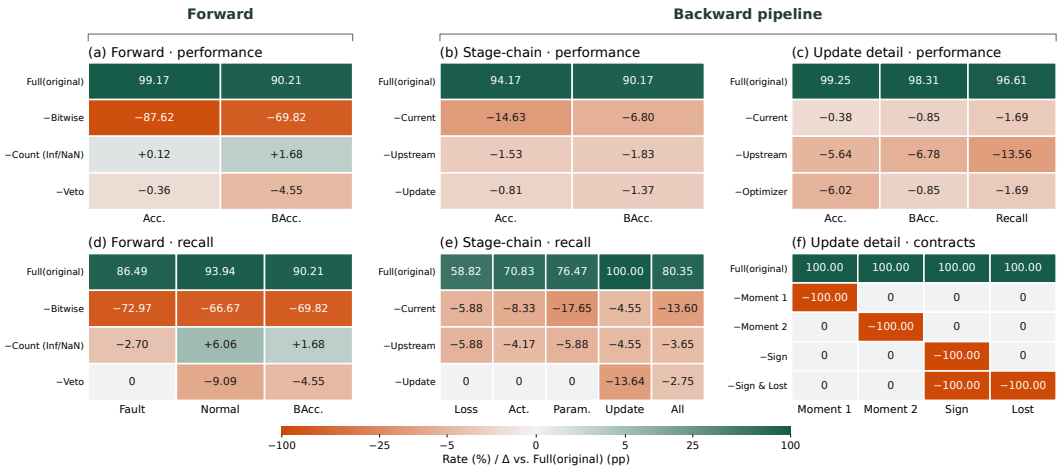

Figure E.3: **Component contributions under frozen configurations.** (a,d) Forward; (b,e) complete backward chain; (c) update assessment; (f) local optimizer checks. Full(original) gives absolute percentages; removal rows give percentage-point changes. Accuracy is pair-weighted; balanced accuracy and recalls are group-weighted. Panel-specific cohorts and branch definitions in Appendix E.4.

from 128 groups. Full(original) reports absolute percentages; removal rows report percentage-point changes from the corresponding full configuration. Accuracy is pair-weighted, and balanced accuracy and recalls are group-weighted. Undecided cases remain in the denominator. Count (Inf/NaN) denotes the count trigger selected in each frozen fold; Veto denotes its alignment-veto branch. Current and Upstream denote current-stage and upstream evidence. Removing Update disables update-local evidence while retaining upstream evidence. Moment 1/2, Sign, and Lost denote first-/second-moment residual, update-sign disagreement, and lost-update checks. The joint Sign & Lost removal exposes their overlapping detection coverage.

Figure 4. Forward calibration with a fixed metric inventory and decision structure. Each budget denotes labeled support pairs per fold; all models use the same 840-pair, five-fold group-isolated evaluation union. (a) Group-balanced accuracy; (b) normal and fault recall; (c) the number of 20 support-set draws reproducing full-data performance. Curves show means and shading indicates P10–P90 across draws. Six pairs are the smallest tested budget at which all 20 draws reproduce full-data performance. (d) Availability of independent matched controls in the historical backward protocol, reported as a protocol audit rather than evidence of backward few-shot calibration.

Figure E.2. Six-shot calibration across 100 new random support-set draws. (a) Successful complete five-fold calibration; (b) successful individual fold fits; (c) exact recovery of full-data performance; (d) group-balanced accuracy for successfully fitted models. Sampling three normal and three faulty pairs from the evidence-complete pool succeeds in all 500 fold fits and reproduces full-data performance in all 100 evaluations. Eligibility uses evidence availability, not metric values or validation outcomes. Unfitted configurations are marked Not fitted rather than assigned zero accuracy.

F Representative Diagnostic Cases

We study two precision changes over 1,000 training steps: router Softmax in Qwen3-30B-A3B and RMSNorm in Qwen3-0.6B. The MoE case connects a localized early discrepancy to subsequent routing behavior. The Dense case complements this analysis by showing how training fit and held-out quality evolve under different precision policies. Figures F.1 and F.2 present the main observations; the tables provide the settings and numerical values needed to interpret them.

F.1 Experimental settings

For Dense, the registered configuration uses global batch size 16 and AdamW with learning rate 2×10^{-5}, β_1=0.9, β_2=0.95, ϵ=10^{-8}, weight decay 0.1, 100 warmup steps, and gradient clipping at 1.0. Apart from the variant identifier, the resolved reference and target configurations differ only in the RMSNorm precision field. Both historical cases retain their registered Qwen2.5-7B tokenizer

Table 5: Experimental settings for the two precision-change cases.

Setting	MoE	Dense
Model	Qwen3-30B-A3B	Qwen3-0.6B
Precision change	Router Softmax: FP32 → input BF16	RMSNorm: FP32 → BF16
Changed scope	Router policy across 48 layers	RMSNorm precision policy
Hardware	8 H200 GPUs	4 H200 GPUs
Training length	1,000 updates	1,000 updates
Input length	512 tokens	512 tokens
Training data	One frozen C4 sequence, repeatedly used	Historical `c4_train_200.jsonl` training input
Seed	42	42
Comparison	Reference and target, each with a deterministic repeat	Reference and target, each with a deterministic repeat
Final evaluation	Common FP32 router; BF16 model	Common FP32 RMSNorm, RoPE and Softmax; BF16 model
Validation input	128 C4 sequences; 65,408 predicted tokens	128 C4 sequences; 65,408 predicted tokens

files. The comparisons therefore concern these paired experimental settings rather than standard model benchmark scores.

F.2 MoE: from a local discrepancy to different routing paths

The first training step provides a particularly informative diagnostic observation. The two paths produce identical first-layer router logits, yet their Softmax probabilities differ. Thus, the immediate input to the candidate computation is held fixed while its output changes. The localization procedure returns `TokenChoiceTopKRouter.forward:softmax`; replaying the original computation with the same logits reproduces the outputs of the two precision paths. The resulting relative L2 difference is approximately 1.67034×10^{-3}. After saving the localization result, comparison with the registered change record confirms that the first-layer instance lies within the modified router policy.

The early observations show why inspecting only expert identities would miss part of the numerical change. The first layer still selects the same experts, but 24.66% of its mixing-weight entries differ. The discrepancy is already present in the continuous routing computation before it changes the discrete expert selection at that layer.

Table 6: First-step, first-layer MoE evidence.

First-step, first-layer observation	Reference–target comparison	Interpretation
Router logits	Exact agreement; relative L2 = 0	Same immediate Softmax input
Router probabilities	Relative L2 $\approx 1.67034\times10^{-3}$; max absolute difference $\approx 3.79324\times10^{-4}$	Numerical discrepancy at Softmax output
Captured top-9 expert indices	Exact agreement	Top-8 identities also agree
Top-8 mixing weights	24.6582% of entries differ; relative L2 $\approx 2.63491\times10^{-3}$	Changed routing weights despite matching identities
Tokens per expert	Exact agreement	First-layer assignment counts agree
Localization output	First-layer router Softmax	Candidate matches the recorded change scope

This is a retrospective localization check: the scan did not read the change record, while the researchers knew the intervention. The result identifies the earliest supported source instance among the saved observations. The diagnostic prefix is a five-step companion run whose identity and first-five-step mathematical scalars match the long run.

For layer–token position i, let R_i and T_i be the two sets of eight selected experts. The **changed-set rate** is the fraction of positions for which $R_i \neq T_i$. The **member replacement rate** averages (8 - —$R_i \cap T_i$—)/8 over positions. A single replaced expert makes a position count toward the

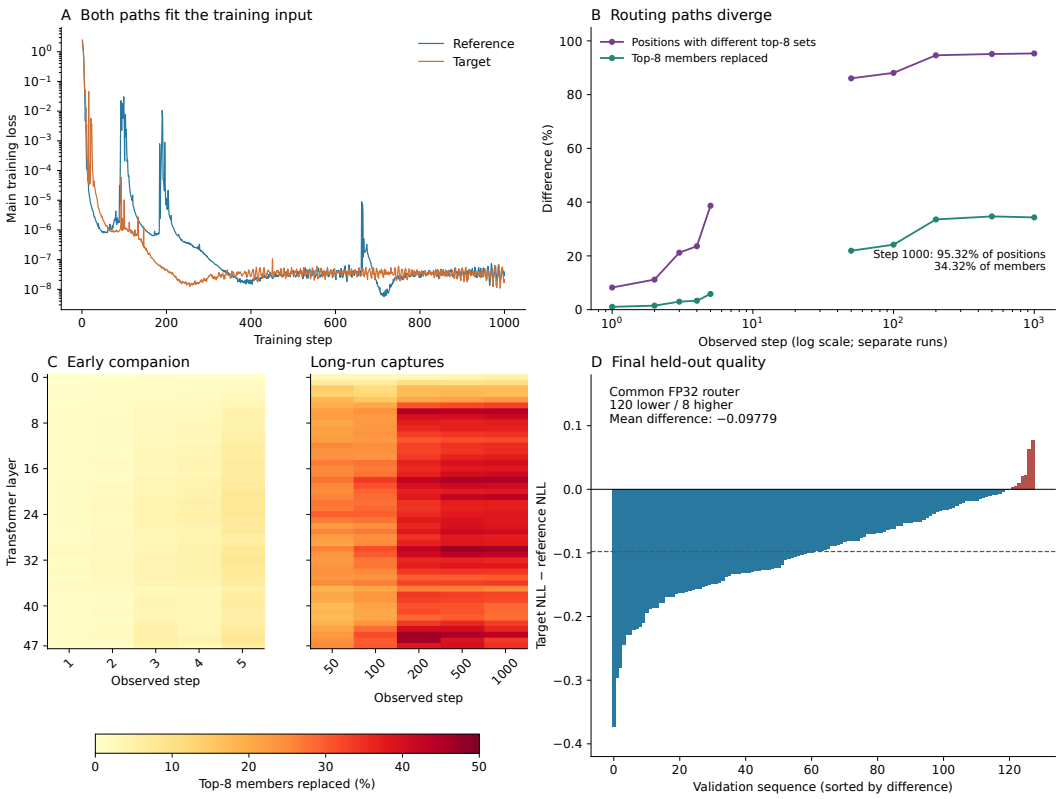

Figure F.1: **Routing divergence despite low training loss.** (A) Both paths fit the repeated training input, with final main losses of 3.00939×10^{-8} and 2.14623×10^{-8}. The logarithmic axis retains the transient loss variation as well as the low-loss regime. (B) The fraction of layer–token positions with different top-8 sets and the fraction of replaced top-8 members describe two aspects of routing divergence. Early companion and long-run observations are drawn as separate segments. (C) Layer-wise replacement rates show where the divergence occurs, using one common color scale. Columns denote sampled steps, not equal time intervals. (D) Final-checkpoint validation uses the same FP32 router policy for both paths. Each bar is one sequence's target-minus-reference NLL; negative values favor the target.

first metric, but contributes only 1/8 to the second. Their distinction explains why 95.32% changed positions at step 1,000 corresponds to 34.32% replaced members.

The following values are recomputed from the saved per-layer routing statistics. The first layer retains its expert set at step 1, while other layers already differ; hence the nonzero all-layer rate at that step.

Table 7: Routing divergence across saved observations.

Capture	Step	Changed-set positions (%)	Replaced members (%)	Layers with changes
Companion	1	8.24	1.05	47 / 48
Companion	2	11.18	1.49	48 / 48
Companion	3	21.20	2.97	48 / 48
Companion	4	23.60	3.32	48 / 48
Companion	5	38.66	5.84	48 / 48
Long run	50	86.07	21.92	48 / 48
Long run	100	88.10	24.17	48 / 48
Long run	200	94.65	33.56	48 / 48
Long run	500	95.11	34.70	48 / 48
Long run	1000	95.32	34.32	48 / 48

At step 1,000, all 48 layers have at least one changed expert set. Mean member replacement is 34.32%, with per-layer rates ranging from 4.35% to 46.68%. Figure F.1 juxtaposes this broad routing divergence with the low losses of both paths. The diagnostic contribution is therefore concrete: it identifies an early source of numerical differences and exposes internal behavior that aggregate training loss does not summarize.

The common-policy final evaluation yields NLLs of 3.28599 for the reference and 3.18820 for the target. Of the 128 validation sequences, 120 favor the target and eight favor the reference. These measurements complete the case by reporting the quality associated with the two training paths, rather than assigning a quality direction from the routing difference alone.

F.3 Dense: training fit and validation behavior

The Dense experiment changes RMSNorm computation from FP32 to BF16 while retaining FP32 RoPE and Softmax. Its training records show nearly overlapping loss trajectories. The first 50 steps have mean recorded losses of 3.31282 and 3.31313; the last 50 steps have means of 0.058587 and 0.059133. The historical held-out evaluations show a different trend: NLL rises from approximately 3.33 at step 50 to approximately 7 at step 1,000.

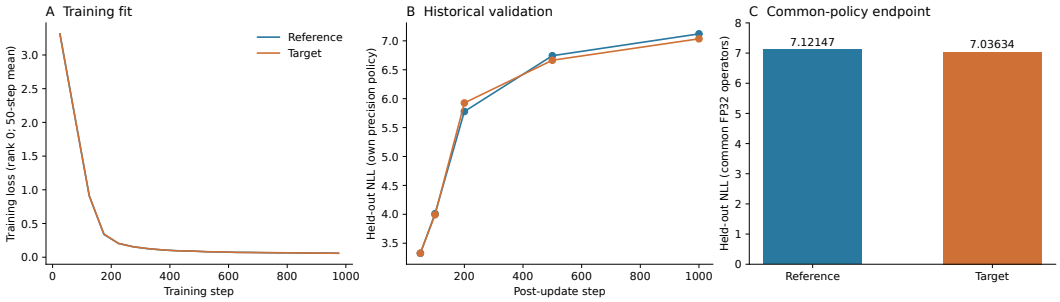

Figure F.2: **Improved training fit alongside increasing validation NLL.** (A) Non-overlapping 50-step means of the recorded rank-0 training loss. (B) Historical held-out NLL at the five saved checkpoints, evaluated under each path's own precision policy. Lines connect measured checkpoints as visual guides. (C) A separate final-checkpoint evaluation uses a common FP32 policy for RMSNorm, RoPE and Softmax. This endpoint comparison is kept separate from the historical curve because the evaluation policy differs.

Table 8: Historical Dense validation under each path's precision policy.

Post-update step	Historical NLL		Target − reference
	Reference	Target	
50	3.32632	3.32730	+0.00098
100	4.01027	3.99266	-0.01761
200	5.77789	5.92564	+0.14775
500	6.74266	6.66438	-0.07828
1,000	7.12035	7.03523	-0.08513

The ordering of the two paths changes across checkpoints even as validation NLL increases on both sides. This makes the Dense example useful alongside the MoE case: training fit, relative quality and internal numerical behavior carry different information. The declining training loss and increasing validation NLL are consistent with repeated fitting to limited training data; the shared validation trend is not attributed solely to the precision change.

Under the common final evaluation policy, reference and target NLLs are 7.12147 and 7.03634. The difference remains when the evaluation operators use the same precision, showing that the endpoint contrast is not solely a consequence of evaluating the two checkpoints with different operator policies. The Dense case provides the longitudinal comparison; the detailed source-localization evidence in this appendix comes from MoE.

F.4 FINAL-CHECKPOINT COMPARISON

Table 9: Final-checkpoint evaluation under common precision policies.

Case	NLL		ΔNLL (T−R)	PPL		Sequences favoring
	Reference	Target		Reference	Target	target / reference
MoE	3.28599	3.18820	-0.09779	26.7353	24.2447	120 / 8
Dense	7.12147	7.03634	-0.08513	1238.2745	1137.2226	90 / 38

NLL is weighted by the number of predicted tokens, and PPL is exp(NLL). Each case uses the same validation input for its reference and target. The deterministic repeats reproduce the per-sequence evaluation results exactly. These are paired descriptive comparisons within each case; differences between the two model rows are not an architecture ranking.

Together, the cases demonstrate how localized numerical evidence and long-horizon measurements complement one another: localization supplies a concrete computation to investigate, while routing traces and task evaluations reveal distinct aspects of subsequent training behavior.

F.5 OBSERVATION PROTOCOL AND SUPPORTING MATERIAL

The MoE scan covers 486 saved forward observations and uses the historical frozen observation rule's bitwise baseline branch with the original localization API. Same-side repeats and same-input replay provide the local controls. This case analysis documents source localization; the full non-elementwise alignment classifier is evaluated separately from this historical observation branch. The registered precision change applies across 48 router layers, while the localization result establishes the first supported instance. The early captures and long-run observations are kept distinct in all plots.

Historical Dense intermediate evaluations retain their own precision policies, whereas the new end-point evaluations use a common policy. MoE has final held-out evaluations but no corresponding intermediate held-out curve. The two cases are paired precision experiments on H200, rather than cross-hardware comparisons.

G TRAINING ERRORS HIDDEN BY LOSS

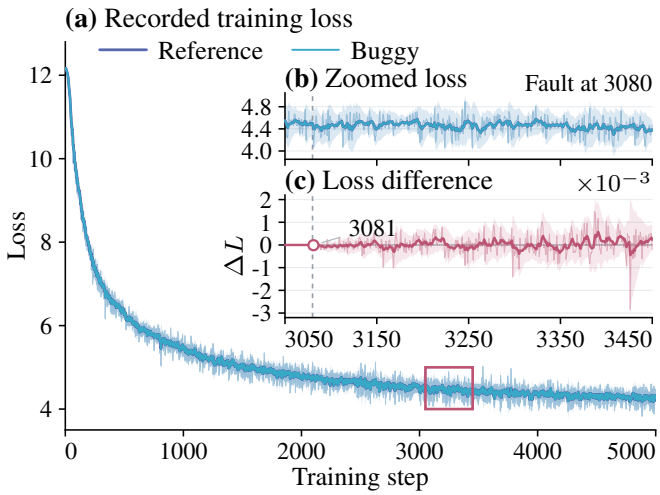

Figure G.1: **Loss can conceal a local training error.** OpGuard's embedding-backward incident, redrawn from its public records (Zhou et al., 2026): (a) recorded training loss; (b) steps 3050–3450; (c) paired difference $\Delta L = L_{\text{buggy}} - L_{\text{reference}}$. Faint curves show every raw observation; solid curves show a normalized causal EMA (decay 0.8), with ± 2 weighted temporal SD bands (not confidence intervals). Statistics use the full history; no points are thinned. At fault step 3080, $\Delta L = 0$; the circle marks the raw difference -1.0014×10^{-5} at 3081.