When Metamorphic Contrast Hurts: A Negative Result for Rotation-Conditioned Detection of Injected Neural-Network Drift

Anonymous Authors

Abstract

Machine-learning models are rarely deployed once and left alone. The engines that run them keep changing underneath: inference software gets upgraded for speed, optimizers get switched on, numerical kernels get swapped out, all while the model file itself stays the same. These upgrades almost never crash. Instead, they can silently nudge a handful of output values just enough to flip a decision that was sitting near a boundary – a quality-control camera on a factory line that stops flagging a scratched part, or a hospital triage tool that quietly drifts on exactly the scans it was already least confident about. Because there is no independent ground truth for what a model's output "should" be after such an upgrade, testers reach for an oracle that does not need one: run the model on an input and again on a lightly transformed copy of it (a 10-degree image rotation, which should not change what the image shows), and compare how much the two software versions disagree on each. It is tempting to subtract one disagreement from the other, on the reasoning that whatever is common to both is background noise and whatever differs is the real signal. We show this reasoning can backfire. Across three public CIFAR-100 classifiers, two pinned ONNX Runtime configurations, and 54 controlled weight-perturbation faults (16,200 faulty and 900 clean held-out cases), the subtraction-based detector caught 75.48% of injected faults versus 82.75% for simply comparing outputs on the unrotated image alone, at the same 0.67% false-alarm rate. A 5,000-replicate bootstrap that redoes the calibration itself, not just the test split, keeps this gap negative throughout (95% interval -8.80 to -3.81 points), and the same pattern holds, more sharply, for pinpointing which layer was perturbed. The mechanism is intuitive in hindsight: a persistent fault shifts behavior similarly on both the original and the rotated input, so subtracting the two disagreements discards exactly the evidence a detector needs to notice anything is wrong. For anyone building automated tests for this kind of silent drift, the practical lesson is concrete: a two-input comparison is not automatically better than a one-input one, and it must be checked against the plain baseline at a matched false-alarm rate before it is trusted to catch the next drifting camera or drifting triage tool. The finding is scoped to the tested substrate of injected weight drift, not a verdict on metamorphic testing as a strategy.

Keywords: metamorphic testing, differential testing, ONNX Runtime, numerical drift, deep-learning testing, negative results, calibration uncertainty, reproducibility

1 Introduction

Picture a small manufacturing line that photographs each part on the belt and runs the image through a neural network to decide whether it passes quality control. The network was trained once, validated carefully, and has been running in production for a year. This spring, the team upgrades the inference engine that executes the network – the software layer, commonly ONNX Runtime, that loads an exported model file and runs it efficiently on the available hardware – to a newer release that promises faster inference, and turns on its graph-optimization option to shave a few more milliseconds off each decision. Nothing crashes. The nightly regression suite, which checks average accuracy on a held-out batch of parts, reports no change. Weeks later, someone notices a specific class of hairline scratches is now passing inspection that used to be caught. Nobody changed the model. The runtime did.

Now picture a second, higher-stakes version of the same story. A hospital's radiology department uses a commercial triage tool that flags chest X-rays likely to show a collapsed lung, so a radiologist reviews the flagged cases first instead of working strictly in queue order. The vendor ships a routine update – a newer build of the same inference engine, with the same optimization flag turned on for speed. The tool's overall sensitivity on the vendor's validation set is unchanged, so the update passes review. But sensitivity is an average over thousands of cases, and averages can hide the kind of narrow, input-selective shift this update introduces: if a handful of borderline scans that used to sit just above the flagging threshold now sit just below it, nobody's dashboard will show it, until a case that should have been prioritized was not. This second scenario is deliberately illustrative rather than a documented incident, but "the model didn't change, only the software running it did" is a property of deployed machine-learning systems generally, not a quirk of any one industry.

This is the general problem of silent numerical drift: an update to the software that executes a model can change a handful of its output values without changing the model itself and without producing any error, exception, or obvious accuracy drop. The same risk applies well beyond factory cameras and hospital triage queues, to any service that treats its model file as fixed while everything around it keeps being upgraded. What makes this hard to test is the absence of an oracle: nobody has, for every possible input, a trusted second source that says what the "correct" output should be after the upgrade. Ordinary regression testing, which checks aggregate accuracy on a fixed labeled set, is not built to catch a rare, input-selective shift like the ones in these two scenarios – it only flags a problem if enough individual cases flip to move the average, and a handful of borderline cases rarely do.

Two established testing strategies attack exactly this oracle-free situation. Differential testing compares two implementations or configurations that are expected to agree with each other, without needing to know the ground-truth answer for either one – if the old and new runtime disagree on an input, something changed, whether or not either answer is externally verifiable on its own. Metamorphic testing instead relates two executions connected by a transformation that should not matter to the task at hand: if rotating a photograph by ten degrees should not change what it depicts, a well-behaved model's behavior on the rotated photograph should relate predictably to its behavior on the original, even without knowing either "correct" answer in isolation. Deep-learning testing has used both ideas: early framework validation applied metamorphic relations [2], while cross-implementation disagreement became a practical oracle for guided neural-network testing [3]. These two ideas are compatible, and it is natural to combine them: run both the old and new configuration on a source input and on a transformed follow-up input, and build a single detector out of all four resulting outputs.

That combination creates a design choice that is easy to get wrong. A neural network's raw output, before it is turned into a class label, is a vector of numbers called logits – one per possible class, with the largest one indicating the network's preferred answer. Suppose the new-minus-old difference in these logits on the source input is a vector e_o, and the corresponding difference on the rotated follow-up input is e_t. A detector that only looks at the source input keeps the size of e_o. A rotation-conditioned metamorphic detector instead looks at the size of the difference e_t - e_o – it subtracts the two errors before deciding whether anything is wrong. Subtracting is appealing: whatever is common to both errors might be a harmless side effect of comparing two valid configurations, and whatever is left over might be specific to the fault. But the same subtraction has an unavoidable dark side. If an upgrade shifts a model's behavior in a way that is largely input-independent – the same weight, used the same way, regardless of which image is fed in – then e_o and e_t point in similar directions, and subtracting them removes most of the very signal the detector is trying to expose. A transformation can be entirely semantically valid while the arithmetic built on top of it quietly cancels the fault it was meant to catch.

This distinction matters in practice, not just in theory. A detector that uses a rotated follow-up image costs more to run than one that does not: it requires an extra inference pass, a choice of transformation, and its own calibration – the process, described in full below, of deciding how large a discrepancy has to be before

it counts as an alarm. If that extra cost is compared against a plain single-input baseline only at a loose, uncalibrated tolerance, an apparent improvement may really be an artifact of the larger execution budget, or of comparing two detectors at different false-alarm rates rather than the same one. A fair evaluation keeps these pieces separate: is the transformation valid, is the detector calibrated, is the execution budget matched, and is the comparison unit handled correctly.

Calibration matters especially at the low false-alarm rates a real deployment would demand; nobody wants a factory line or a hospital queue interrupted by a stream of false alerts. Each detector's alarm threshold here is set from only 300 held-out clean images per model, at the value that would flag just 1% of them. Treating that threshold as a known constant when estimating uncertainty is optimistic, since it would come out slightly differently with a different 300 images. To address this, the analysis adds a nested resampling procedure – explained in the Methods section – that redraws the calibration images, re-estimates every threshold from scratch, and separately redraws the held-out test images, for five thousand repetitions, directly testing whether the negative result survives realistic calibration uncertainty, not just test-set noise.

The experiments use three public, pretrained convolutional networks trained on CIFAR-100, a widely used labeled image dataset, executed under two pinned CPU configurations of ONNX Runtime – an older release with graph optimization turned off, and a newer release with it turned on, mirroring the kind of upgrade a production team might make. A frozen protocol, written down before any results were seen, specifies a disjoint calibration/test split, a 10-degree rotation, three families of controlled weight perturbations injected at an early, a middle, and a late convolution layer, two severities, independently calibrated alarm thresholds, and a resampling scheme that keeps every fault case tied to the source image it came from, for 54 distinct fault configurations and 16,200 held-out faulty test cases. Because the faults are deliberately injected into a known convolution, the study also has exact ground truth for a second question: not just whether a detector notices something changed, but whether it can point to which layer changed. This design is a stand-in for the kind of persistent numerical shift a runtime or optimizer upgrade can introduce, not a random sample of historical bugs.

The central question is whether rotation-conditioned subtraction actually improves on the plain, single-input comparison it is built from, once both are compared honestly at the same false-alarm rate – and whether the same holds for localizing which layer changed. Four smaller, exploratory follow-up checks are also reported: whether giving the plain comparison the same two executions closes the gap; whether the result is peculiar to one rotation angle or interpolation method; whether the errors point in the directions the cancellation explanation predicts; and whether separating the runtime-version and optimization-flag factors changes the picture.

The evidence supports five findings. First, at matched false-alarm rates, the subtraction-based detector catches noticeably fewer of the injected faults than the plain comparison: 75.48% versus 82.75%, both at a 0.67% false-alarm rate on clean data. Redoing the calibration itself five thousand times keeps this gap negative throughout, from -8.80 to -3.81 percentage points at the 95% level, while the corresponding gap in false-alarm rate spans zero. Second, the same pattern is sharper for localization: subtraction correctly names the perturbed layer as its top guess only 15.15% of the time, against 45.15% for the plain comparison. Third, giving the plain comparison the same two executions – without subtracting them – reaches 84.52% detection at a slightly higher 1.11% false-alarm rate, showing the extra input is not the problem; discarding the common signal by subtracting is. Fourth, the negative gap persists across four rotation angles and interpolation methods. Fifth, the errors behave the way the cancellation explanation predicts – more aligned on cases the plain method alone catches, less aligned on cases only subtraction catches – though this is a consistency check, not independent proof of the mechanism.

This is a bounded, negative result, not a verdict on metamorphic testing as a family of techniques. It says nothing about settings where no second implementation exists to compare against at all, where a transformation reaches genuinely new behavior rather than relating two existing executions, or where the underlying faults look nothing like the injected weight perturbations studied here. What it does establish is a

concrete burden of proof for anyone building a two-input drift detector out of already-available differential errors: before trusting a version that subtracts them, check it against the plain, undifferenced version at the same false-alarm rate, with calibration uncertainty and execution cost made explicit. For the factory floor and the radiology queue alike, that burden of proof is the difference between a test suite that would have caught the drift, and one that would not have.

2 Related Work

Metamorphic testing begins from the observation that a program may lack an affordable test oracle while still admitting necessary relations among multiple executions. Segura et al. organize relation construction, test generation, and applications across domains [1]. Their framework highlights two separate stages, both relevant here. First, a transformation and the relation it should preserve must be semantically justified – rotating an image should not change what it depicts. Second, an executable rule must decide whether observed outputs actually violate that relation. This paper concentrates on the second stage. It does not claim that arbitrary CIFAR-100 rotations are universally invariant; it asks whether subtracting two already-available differential-error vectors yields a stronger, properly calibrated detector than simply keeping either vector's size. This distinction – between whether a relation is semantically sound and whether the numerical rule built on top of it is a good decision procedure – is the load-bearing idea of the whole paper, worth stating plainly because it is easy to conflate the two questions when reading a single result out of context.

Ding et al. offered an early application of metamorphic testing to validating a deep-learning framework [2]. DeepHunter later combined semantics-preserving image mutations with coverage guidance to generate tests for deep neural networks [5]. Both studies show the value of a follow-up execution when exact labels or outputs are hard to assert. Their goals are broader than the narrow scoring question isolated here: a transformation can improve testing simply by reaching new activation patterns or triggering an input-selective defect, independent of whether subtraction is the best way to combine reference and candidate discrepancies. Holding inputs, models, and fault configurations fixed separates scoring effectiveness from test-generation effectiveness – a technique can be excellent at generating interesting test cases while still being a poor way to score the ones it generates.

Differential testing of neural networks supplies the other half of the hybrid oracle used here. DeepXplore cross-references multiple systems and uses their disagreement, together with neuron-coverage guidance, to drive white-box input generation [3]. The direct-L2 baseline in this paper follows the same empirical-oracle principle, but at a runtime-configuration boundary rather than across independently trained implementations. The metamorphic contrast studied here does not eliminate differential testing – both vectors it subtracts, e_o and e_t, are themselves candidate-minus-reference differential errors. The contrast should not receive methodological credit merely for consuming a follow-up input: it uses two direct discrepancies and throws away their common component, and the experiment tests whether that discard actually improves discrimination once both are calibrated on clean data.

CRADLE is the closest prior system for combined detection and localization [4]. It performs cross-backend validation and traces anomalous behavior toward a likely faulty library function. Its localization target differs materially from the one used here: this study perturbs a known convolution layer and ranks instrumented convolution outputs against that known origin, whereas CRADLE diagnoses naturally arising cross-backend inconsistencies down to library functions. Controlled injection gives exact ground-truth layer labels and supports a clean paired ranking comparison, but cannot establish which library function is actually at fault the way CRADLE's diagnostic setting can. The two settings are complementary: CRADLE answers "which function is broken" for defects that already occurred in the wild, while this paper answers "does a scoring rule find the right layer" for a defect whose location is known by construction.

DiffChaser searches for disagreements between an original and an optimized neural-network variant

[6]. This is close in spirit to the runtime/configuration contrast studied here, but its core contribution is disagreement discovery through search: it finds inputs where two configurations diverge sharply or in a decision-relevant way. This paper's inputs and mutations are fixed by design; it evaluates how already-observed discrepancies are converted into pass/fail decisions. Finding disagreement well does not by itself establish which norm or combination is the best way to turn that disagreement into a calibrated alarm.

Model- and architecture-generation systems expand the reachable test space in a different direction. LEMON guides model generation toward variants likely to expose cross-library inconsistencies [7]. Audee searches model structures, parameters, weights, and inputs, and works toward localizing framework inconsistencies [8]. Muffin extends differential testing into the training phase and generates architectures for cross-library comparison [12]. These systems address breadth and trigger generation, while this paper holds the pretrained models and fault configurations fixed. Audee is especially relevant because it moves from disagreement detection toward identifying influential layers or parameters; the simple activation sketches used here are not a substitute for its causal analysis, which intervenes on a model directly rather than merely observing its outputs.

GRIST addresses a complementary part of the same broad problem: it uses gradient back-propagation to synthesize inputs that expose reachable NaN or infinity failures in deep-learning programs [9]. The matrix studied in this paper contains finite, non-crashing discrepancies on a fixed sample of images; it neither searches for exceptional values nor says anything about GRIST's ability to trigger them. A detector that performs poorly against persistent, finite weight drift could behave quite differently against discontinuous or strongly input-selective failures.

Compiler and computation-graph testing work at a different level of abstraction. MT-DLComp applies semantics-preserving model transformations to deep-learning compiler testing [10]. EAGLE generates equivalent computation graphs and compares execution within a single library [11]. Model-level metamorphic testing constructs structural model variants intended to preserve semantics across interface combinations and runtime metrics [17]. These transformations act on models or computation graphs rather than modest image rotations, and can embody stronger, operator-specific equivalences whose error-direction behavior may differ from the rotation studied here. None of these systems is contradicted by this negative result – their relation inventories, generated tests, and bug yield are outside its scope. This paper's ablation begins only after a reference and candidate execution have already been produced, and asks what to do with the two discrepancies that result.

Compiler fuzzers focus on producing effective tests in the first place. Tzer combines coverage guidance with mutation of tensor intermediate representations and optimization passes [13]. NNSmith generates diverse valid models and seeks numerically well-behaved inputs before running differential compiler checks [15]. Both reduce wasted executions on invalid or pathological programs while still reaching states likely to reveal real defects. This paper's design begins after a valid execution pair and a controlled perturbation already exist; it isolates the downstream decision rule once raw disagreements must be converted into alerts within a defined false-alarm budget.

Empirical bug studies motivate the broader problem without validating any particular detector. Wang et al. characterize the causes and symptoms of numerical bugs in deep-learning programs [14]. Jajal et al. analyze interoperability failures in ONNX model converters, including semantically incorrect, non-crashing outcomes [16]. This evidence supports paying explicit attention to silent numerical and semantic divergence in deployed machine-learning stacks – exactly the concern illustrated by the factory-camera and radiology scenarios in the Introduction. It does not imply that deterministic weight scaling, Gaussian weight noise, and quantization are a representative sample of historical defects; these studies are used here for motivation, not as indirect evidence about the detectors' recall on real-world bugs.

The closest substrate-specific diagnostic work is DiTOX, which targets fault detection and pass-level localization inside the ONNX Optimizer [19]. Its optimized-versus-unoptimized comparison makes it a natural foundation for a future evaluation against historical optimizer defects. The primary configuration

studied here also contrasts an optimized and an unoptimized runtime state, but the injected faults are convolution-weight modifications and the ground truth is the originating convolution layer – a design that enables an exactly labeled experiment, whereas DiTOX's pass-oriented diagnosis is closer to the causality a maintainer cares about when triaging a real bug report. A stronger continuation should combine clean controls and defect-specific relations with patch- or maintainer-derived localization labels, rather than treating an injected convolution as a stand-in for them; the historical-defect corpus reported later in this paper is a first step toward that continuation.

Formal equivalence checking offers a different answer to the same underlying concern. Volta checks the equivalence of structured, optimized machine-learning GPU kernels against reference kernels [18]. Within its supported language and hardware model, a proof can be stronger than any amount of sampled execution: a verified kernel is guaranteed correct on every input, not merely the ones tested. The setting studied here is empirical and end-to-end instead: it observes public networks under pinned runtime configurations and calibrates detectors from clean samples. Formal and empirical approaches remain complementary – a deployment stack may fall outside a verifier's supported language, while sampled testing can never establish equivalence for every possible input.

Across this literature, three separate objectives are worth distinguishing, because conflating them is a common source of overclaiming. Generation produces inputs, models, graphs, or compiler states likely to exercise a defect. Oracle construction specifies a relation expected to hold among executions. Scoring and calibration convert raw observations into decisions within a defined false-alarm budget. Prior systems make substantial contributions to the first two objectives; the gap this paper addresses is the third: whether subtracting two errors inside a plausible hybrid oracle adds discriminative information beyond what the two errors already carried on their own.

The relevant geometry, while elementary, is the mechanism the rest of the paper repeatedly returns to. Let e_o = c_o - r_o and e_t = c_t - r_t be the candidate-minus-reference logit errors for the source and the transformed input, where c denotes the candidate (upgraded) configuration's output and r the reference (older) configuration's output. The squared size of the subtracted contrast satisfies ——e_t - e_o——^{2} = ——e_t——^{2} + ——e_o——^{2} - 2·——e_t——·——e_o——·cos(theta), where theta is the angle between the two error vectors, running from 1 (identical direction) through 0 (unrelated) to -1 (opposite). When the two errors point in a similar direction, subtracting them shrinks the result; when perpendicular, the squared contrast approaches the sum of the two squared sizes; when opposite, subtracting amplifies the result. Relative size matters too – even two errors pointing in exactly the same direction do not fully cancel if one is much larger. Subtraction can help when clean-data errors happen to be more aligned than faulty-data errors, or when a fault reverses direction or triggers selectively between the two inputs. It can hurt when a persistent fault produces similar errors on both inputs regardless of the transformation – exactly the behavior a weight perturbation is expected to produce, since the same modified weight is used in both executions.

This identity also sharply bounds what any explanation built on it can claim. The angle between the two errors and the relative size of the subtracted contrast are calculated from the same two vectors, so an association between them is algebraically expected rather than an independent discovery. Observing it can show the evaluated fault matrix sits in a cancellation-compatible regime, and describe which cases the two detectors disagree on. It cannot establish causation or show that error alignment predicts performance on faults or relations outside this study; establishing that would require evaluating the alignment rule on faults withheld from its formulation, a continuation this paper leaves for future work.

3 Methods

Study status and chronology. The primary protocol was recorded before execution, in a version-controlled document that fixes the research question, models, dataset split, runtime configurations, transformation,

fault matrix, detector formulas, calibration rule, resampling unit, bootstrap count, multiplicity correction, stopping rule, and the conjunctive superiority criterion that would count as a positive result. It was not deposited with an external registry or independent timestamping service, so this paper uses the phrase ”locally recorded protocol,” not ”preregistered.” A later cost-matched amendment, described below, was formulated only after seeing the primary result and is treated throughout as post-primary and exploratory, as are the rotation-sensitivity, runtime-factor, and error-geometry analyses.

Models and data. The study uses three public, pretrained CIFAR-100 image classifiers exported to the ONNX format: ResNet-20, MobileNetV2 (width multiplier 0.5), and VGG-11 with batch normalization. CIFAR-100 is a standard, publicly available benchmark of small color photographs, each labeled with one of one hundred everyday object and animal categories; it was chosen because it is small enough to run three full models through 54 fault configurations and two runtime pairs in feasible compute, while still being a real photographic classification task rather than a toy synthetic one. These three architectures contain 21, 52, and 8 convolution layers respectively, giving a range of instrumentable localization targets, since architectures with very different numbers of layers let the localization experiment test whether a scoring rule still finds the right one when there are more candidates to distinguish. On a 600-image reference sample, their original-input top-1 accuracies – the fraction of images for which the model’s single most confident guess is correct – are 72.83%, 74.67%, and 73.33%. Accuracy describes the substrate; it is not used to select models or set thresholds. Model files, the CIFAR-100 data source, and every derived sample are checksummed and version-pinned in the accompanying reproducibility package.

A fixed-seed, class-balanced round-robin procedure selects 600 unique CIFAR-100 test images and assigns 300 to calibration and 300 to a disjoint held-out evaluation set, with zero overlap verified in the sample manifest. The same 600 source images are reused across all three models and all 54 fault configurations, which is why source image, not an individual model/fault row, is treated as the unit of statistical resampling throughout: results from the same underlying photograph are correlated with each other in ways a naive per-row analysis would understate, so every uncertainty estimate in this paper resamples whole images, never individual rows.

Transformation. Each held-out image x is paired with a rotated counterpart T(x), obtained by an in-plane rotation of 10 degrees using bilinear interpolation, a standard, smooth method for computing pixel values at the new, rotated positions. The study assumes this modest rotation preserves the CIFAR-100 class label, but no human adjudication of individual pairs was performed. As an indirect check, the reference model’s own top-1 prediction stays the same across the pair for 69.17% of ResNet-20 inputs, 71.83% of MobileNetV2 inputs, and 73.17% of VGG-11-BN inputs, averaging 71.39% – a diagnostic of model stability under the transformation, not proof of semantic validity for every image.

Runtime configurations. The reference configuration runs CPU ONNX Runtime 1.20.1 with graph optimization disabled; the candidate configuration runs CPU ONNX Runtime 1.29.0 with graph optimization enabled, mirroring a realistic ”upgrade and turn on optimization” production change. Because both the version and the optimization flag change together, the primary estimand is the difference between these two complete configurations, not the causal effect of the version upgrade alone. A post-primary factor-isolation analysis executes both versions with optimization disabled and enabled separately: changing the runtime version at fixed optimization produces zero measurable difference in outputs, while enabling optimization produces the same small numerical discrepancy (order 10ˆ-5) regardless of version, with zero prediction disagreement – localizing the clean-input numerical envelope to the optimization flag, though the full fault matrix was not repeated across all four factorial cells.

Fault construction. For each model, the protocol selects one early, one middle, and one late convolution layer. It then applies three families of weight perturbation – multiplicative scaling, additive Gaussian noise, and symmetric weight quantization, the last similar to what a production deployment might do to save memory or speed – each at two severities. The full combination of three models, three layer locations, three fault families, and two severities yields 54 distinct configurations, each evaluated on all 300 held-out images,

producing 16,200 faulty held-out test cases; the modified convolution, its family, severity, and generating parameters are recorded for each. Fault construction never uses held-out detector outcomes to choose or tune parameters.

Because a fault is injected into a known, specific convolution, that convolution supplies exact localization ground truth. This is a deliberate construct simplification, not a claim about real defects: an altered convolution is not the same thing as a compiler pass with a downstream symptom, and the layer that originated a fault is not automatically the most useful place to look for a repair. Without this simplification, there would be no way to know, for a real historical bug, exactly which layer was at fault, and the localization comparison would have nothing to check its rankings against.

Detectors. Write r_o and c_o for the reference and candidate model's output logits on the source image, and r_t and c_t for the same pair of outputs on the rotated image. Define the source error e_o = c_o - r_o and the transformed error e_t = c_t - r_t, exactly as introduced in the Introduction. The direct detector's score is the size of the source error alone, ——e_o——, the ordinary Euclidean length of the vector. The metamorphic contrast studied as the primary candidate is the size of the difference between the two errors, ——e_t - e_o——. A mixed absolute/relative tolerance score, representative of the ad hoc thresholds practitioners often reach for first, takes the largest per-output value of —e_o— divided by a small absolute-plus-relative denominator. Larger scores indicate stronger evidence of drift for all three detectors.

Each detector is calibrated separately for each model, using only that model's 300 clean calibration images: its alarm threshold is set to the empirical 99th percentile of clean scores – in plain terms, set just above where all but the top one percent of ordinary, fault-free scores fall, so the detector should only rarely go off on data with nothing wrong. Thresholds are never adjusted using faulty cases, and a nominal 99th-percentile threshold does not force an exactly 1% false-alarm rate on an independent held-out sample – realized false-alarm rates are measured and reported directly, never assumed.

Cost-matched detector. To rule out the possibility that any advantage for the metamorphic contrast is just an artifact of using an extra execution, a post-primary detector, the two-input direct maximum, takes the larger of ——e_o—— and ——e_t—— – the same two executions the metamorphic contrast consumes, without subtracting them. It is calibrated the same way. Because its realized false-alarm rate differs slightly from the metamorphic contrast's, this comparison controls for execution budget but is not an exactly matched-rate comparison, and is reported as exploratory rather than confirmatory for that reason.

Activation instrumentation and localization. Every convolution layer's activation – the intermediate values a network computes on the way to its final answer – is instrumented with a compact per-channel summary: spatial mean, spatial standard deviation, and maximum absolute value, down-sampled to at most 64 values per layer if larger. For layer l, let a_o,l be the candidate-minus-reference discrepancy in this summary on the source image and a_t,l the same quantity on the rotated image. The direct localization score for a layer is ——a_o,l——; the metamorphic localization score is ——a_t,l - a_o,l——. Each layer's clean 99th-percentile score normalizes its raw score before ranking, so layers with naturally larger activations are not unfairly favored. For every fault case, the analysis records whether the truly perturbed layer is ranked first, in the top three, and its reciprocal rank. A cost-matched activation score, the larger of ——a_o,l—— and ——a_t,l——, parallels the cost-matched detection baseline. These are compact sketch-ranking methods, not full-tensor, gradient-based, or intervention-based causal attribution, and the Limitations section returns to what that restricts the localization claims to mean.

Statistical analysis and decision rule. The frozen decision rule required a positive, source-image-clustered bootstrap confidence interval for the metamorphic contrast's detection advantage over both the direct-L2 baseline and the mixed-tolerance baseline, with a Holm correction for testing the two comparisons together. A bootstrap, for readers unfamiliar with the term, estimates how much a result would vary if the study were run again, by repeatedly re-drawing (with replacement) from the data collected and recomputing the result each time. An original 500-replicate bootstrap resamples the 300 held-out source images (keeping every model, layer, family, and severity combination tied to its source image), while holding the calibration-derived

thresholds fixed at their originally estimated values – valid only for held-out-image variation conditional on the specific calibration sample drawn, not calibration uncertainty itself. A second, nested 5,000-replicate bootstrap instead redraws the calibration images with replacement in every replicate, re-estimates every threshold, independently redraws the held-out images, and recomputes every outcome, directly incorporating calibration and threshold-estimation uncertainty alongside held-out-image variation.

Sensitivity analyses. Four post-primary conditions, each using a smaller 150-calibration/150-test split, repeat the same scores and threshold rules under a 5-degree bilinear rotation, a 20-degree bilinear rotation, a 10-degree nearest-neighbor rotation, and a 10-degree bicubic rotation. These conditions test whether the qualitative result depends on the specific angle or interpolation method; they do not introduce a different transformation family or model class.

Geometry analysis. For every case with nonzero source and transformed errors, the analysis computes the cosine of the angle between e_o and e_t and the ratio of the subtracted error's size to the sum of the two individual sizes, then compares these quantities between cases the plain detector alone catches and cases the metamorphic detector alone catches. Because these quantities are algebraic functions of the same two error vectors used to build the detectors, this is an expected consistency check rather than an independent test of the underlying mechanism, as already flagged in the Related Work section.

Historical issue corpus. A separate, smaller corpus attempts to reproduce publicly reported ONNX Runtime and PyTorch issues whenever an executable, issue-specific oracle and a compatible software environment can be assembled. Fourteen independent issues are attempted through fifteen configurations, with duplicate compatibility probes for the same underlying issue counted once. This corpus was not built with shared clean controls or matched false-alarm-rate scoring, so it establishes only that the reproduced defects are real and reachable, not how well any detector here would perform on them. The primary experiment is separately rerun end-to-end on the same machine, and the raw detection scores, localization ranks, and generating manifests from the two runs are compared for exact, byte-for-byte agreement – not a substitute for independent replication on different hardware, a distinction the Limitations section makes explicit.

Algorithm: Calibrated drift-detection decision rule

Illustrates, at a high level, how each of the three detectors in this study turns a raw discrepancy score into a pass/fail alarm using only clean data for calibration – the same recipe is applied independently to the metamorphic, direct, and mixed-tolerance scores for each model.

1. Given a model and a chosen score function (for example, the size of the source-input error), compute that score on each of the 300 clean calibration images for this model.

2. Sort the resulting 300 clean scores and set the alarm threshold to the smallest score at or above the 99th percentile of this sorted list.

3. For a new test case (clean or faulty), compute the same score function from its reference and candidate outputs.

4. If the test case's score exceeds the threshold, flag it as drift; otherwise, pass it.

5. Record the case's true label (clean or faulty) alongside the flag to compute the realized detection rate and false-alarm rate after the fact – the nominal 99th percentile does not by itself guarantee a 1% realized false-alarm rate on held-out data.

6. Repeat independently for each detector and each model; never use faulty-case scores to choose or adjust the threshold.

Algorithm: Nested calibration-uncertainty bootstrap

Shows how the study estimates uncertainty that accounts for the calibration sample itself being finite, not just the held-out test sample – the procedure behind the widened, more honest confidence intervals reported in the Results and used throughout the Discussion.

1. Repeat the following steps 5,000 times to build a distribution of outcomes.
2. Draw a new set of 300 calibration source images, with replacement, from the original calibration pool.
3. Using only this redrawn sample, re-estimate every detector's 99th-percentile alarm threshold from scratch, separately for each model.
4. Independently draw a new set of 300 held-out source images, with replacement, from the original held-out pool, keeping every fault case tied to its source image.
5. Score every fault and clean case in this redrawn held-out set using the freshly re-estimated thresholds, and record each detector's detection rate and false-alarm rate.
6. Compute the metamorphic-minus-direct difference in detection rate and in false-alarm rate for this replicate.
7. After all 5,000 replicates, report the mean, median, standard deviation, and 95% interval of each difference across replicates as the calibration-aware uncertainty estimate.

4 Results

Primary detection at realized thresholds. The 900 held-out clean cases and 16,200 held-out fault cases were evaluated using thresholds learned exclusively from the disjoint calibration set. Metamorphic contrast detected 12,228 fault rows (75.4815%); direct L2 detected 13,406 rows (82.7531%). Each flagged exactly 6 of 900 clean cases, an identical observed false-alarm rate of 0.6667% for both. The observed metamorphic-minus-direct detection difference was -7.2716 percentage points, at an observed false-alarm-rate difference of zero – so on this particular split, the plain detector's advantage is not bought by a looser false-alarm budget. Table 1 and Figure 1 summarize these primary rates alongside the mixed-tolerance baseline and two threshold-free summaries, AUROC and AUPRC – standard scores for a detector's overall discrimination quality across every possible threshold, both ranging from 0 (worst) to 1 (best), with AUPRC weighting performance at the stricter, low-false-alarm end a real deployment would use. Table 1 makes the ordering visually immediate: metamorphic contrast is clearly below direct L2 on every column, including the two threshold-free summaries that do not depend on where the 99th-percentile cutoff happened to land.

The original 500-replicate held-out-image bootstrap produced a narrow detection-difference interval of [-7.64, -6.87] percentage points. That interval is correct for its limited question – variation across held-out images at fixed, already-estimated thresholds – but it should not be read as the overall uncertainty in the comparison, because it holds the calibration sample and its derived thresholds fixed.

The nested bootstrap changes this picture substantially. Across 5,000 replicates that redraw the calibration images, re-estimate every threshold, and independently redraw the held-out images, the metamorphic-minus-direct detection-difference distribution has a mean of -6.63 points, a median of -6.83 points, a standard deviation of 1.32 points, and a 95% interval of [-8.80, -3.81] points, with the most extreme replicates ranging from -10.23 to -2.53 points. Every one of the 5,000 replicates is negative. Threshold-estimation uncertainty widens the interval by more than an order of magnitude relative to the fixed-threshold version, but it never reverses the direction of the result. The corresponding individual detection-rate distributions clarify why:

metamorphic contrast has a nested mean of 75.65% (95% interval [74.65%, 76.75%]), while direct L2 has a nested mean of 82.28% (95% interval [79.31%, 84.64%]) – direct L2's rate is more sensitive to exactly which calibration images set its threshold, yet even its lowest plausible value stays above metamorphic contrast's highest plausible value.

The corresponding false-alarm-rate comparison tells a different story worth dwelling on, since it illustrates a subtlety a less careful analysis could miss. The nested metamorphic-minus-direct false-alarm-rate difference has a mean of +0.08 points and a 95% interval of [-1.00, +1.11] points – an interval that spans zero, unlike the detection-rate difference. The two detectors matching exactly at 0.67% false-alarm rate in the primary split is therefore a property of that particular split, not evidence the two detectors have systematically identical false-alarm behavior under repeated calibration; the detection-rate gap is the robust finding, while the false-alarm-rate parity is not something a reader should expect to reproduce exactly on a fresh sample.

Mixed tolerance, included as a stand-in for the ad hoc thresholds practitioners often use first, detected only 67.10% of faults at a higher 1.22% false-alarm rate, with an AUROC of 0.878 against 0.921 for metamorphic contrast and 0.954 for direct L2 (Table 1). Metamorphic contrast beats this weaker baseline by 8.38 percentage points at the realized thresholds, and the nested bootstrap confirms a positive interval, [+6.09, +10.00] points. The frozen decision rule required superiority over both baselines together; beating the weaker mixed-tolerance detector cannot offset the deficit against direct L2.

Detection results were consistent, not driven by a single model: direct L2 outperformed metamorphic contrast on every architecture, by 5.00 points on MobileNetV2 (75.22% versus 70.22%), 8.02 points on ResNet-20 (85.15% versus 77.13%), and 8.80 points on VGG-11-BN (87.89% versus 79.09%). The gap also depended heavily on how strong the fault was. At the two saturating extremes – high-severity Gaussian noise, both severities of quantization, and high-severity scaling – both detectors reached 100% detection at every layer location, so these conditions cannot distinguish the two scoring rules; a fault obvious to any reasonable detector says nothing about which detector is more sensitive near the boundary of what is detectable at all. The interesting cases sit closer to that boundary: for low-severity scaling injected at the late convolution, direct L2 still detected 62.11% of cases while metamorphic contrast detected only 4.22%, a 57.89-point gap that accounts for a large share of the aggregate deficit. The pattern is a general sensitivity loss near the boundary, not a wholesale failure to detect large perturbations – which is the practical situation that matters most, since the drift scenarios motivating this study are precisely the small, boundary-crossing kind of shift rather than a catastrophic, obviously-broken one.

Layer localization. Direct activation scoring correctly identified the perturbed convolution as its top-ranked guess in 45.15% of the 16,200 cases; metamorphic activation subtraction did so in only 15.15% – a 30.00-point deficit, larger and more consistent than the detection-rate gap (Table 3, Figure 2). Top-3 recovery followed the same pattern, 56.79% against 29.40%, as did mean reciprocal rank, 0.540 against 0.288: subtraction trails direct scoring by roughly a factor of three at the strictest, top-1 level. This deficit was not uniform across models: direct scoring reached 78.22% top-1 accuracy on the eight-convolution VGG-11-BN instrumentation, but only 24.17% on the 52-convolution MobileNetV2 instrumentation, since more candidate layers make correct localization harder for any method. Metamorphic activation scoring trailed direct scoring in every model, by an even wider relative margin than in detection.

Compute-matched detection. A natural objection is that metamorphic contrast is simply being penalized for how it uses its extra execution, not for using an extra execution at all. The two-input direct maximum – the larger of the two direct errors, without subtracting them – tests this directly. It detected 84.52% of faults at a 1.11% false-alarm rate, comfortably ahead of metamorphic contrast's 75.48% at 0.67% (Table 2). Because the two detectors' realized false-alarm rates differ, this is not a strictly matched-operating-point comparison; it is reported as an exploratory check on execution budget rather than a confirmatory result. Its localization analogue reached 45.19% top-1 accuracy – almost identical to the plain, single-input direct localization score of 45.15%, and far above metamorphic activation localization's 15.15%. The extra, rotated input therefore adds essentially nothing when combined by taking a maximum rather than a difference; the loss specifically

traces to subtraction, not to the second execution itself.

Rotation sensitivity. Table 4 reports the four post-primary conditions that vary the rotation angle and interpolation method. In every one of the four conditions, the metamorphic-minus-direct detection gap stayed negative. At a smaller 5-degree rotation, the gap widened to -9.28 points; at a larger 20-degree rotation, it narrowed to -3.57 points, but the metamorphic detector's own false-alarm rate rose to 2.44% (against 1.33% for direct L2), and the reference model's own predictions agreed across the rotated pair only 52.00% of the time, against 71.39% at the primary angle. Nearest-neighbor and bicubic interpolation at the primary 10-degree angle produced gaps of -6.22 and -8.94 points respectively. No condition in this grid reversed the ordering, and localization stayed similarly unfavorable to subtraction throughout, with metamorphic top-1 accuracy ranging from 15.05% to 17.16% against a direct top-1 accuracy of 46.14% in every reduced-split condition – a pattern that supports treating the negative result as a property of subtraction itself, not an accident of the specific rotation chosen for the primary study.

Error geometry. Across the 16,200 primary fault cases, the correlation between the cosine of the angle separating the two error vectors and the size of the subtracted error relative to the sum of the two individual sizes was strongly negative (Spearman rho -0.985), consistent with the algebraic cancellation mechanism described in the Related Work section: cases the plain detector alone catches have a substantially higher mean cosine (0.590) than cases the metamorphic detector alone catches (0.092), a difference of 0.498. Because these quantities are functions of the same two error vectors, this pattern is an expected consistency check on the mechanism, not an independent validation of it. Splitting cases by whether the reference model's own prediction was stable across the rotation did not remove the effect either: among the 11,430 stable-prediction cases, direct L2 detected 82.10% against metamorphic contrast's 74.64%; among the 4,770 unstable cases, the rates were 84.32% and 77.51%. Prediction instability is therefore not the full explanation for the aggregate gap.

Public issue corpus and exact replication. Of fourteen independently attempted public ONNX Runtime and PyTorch issues, ten reproduced under their own issue-specific oracles (3 of 4 ONNX Runtime issues and 7 of 10 PyTorch issues), an issue-level rate of 71.4% (Wilson 95% interval [45.4%, 88.3%]). Four issues did not reproduce and one further configuration was hardware-incompatible; all outcomes, including the failures, are retained rather than discarded. This corpus demonstrates that the reproducer collection recovers a meaningful set of genuine public failures; it does not measure any detector's recall, since these cases lack shared clean controls, matched-false-alarm-rate scoring, or maintainer-derived localization labels. Separately, an exact rerun of the primary 300/300-image experiment on the same machine reproduced identical raw detection scores, localization ranks, and fault-generating manifest, byte for byte, and every automated integrity and claim check passed on both runs. This establishes deterministic replay and artifact integrity on this machine, not independent replication on different hardware or by a different laboratory.

5 Discussion

The central result is negative and has a concrete design implication: on this three-model, 54-configuration matrix, rotation-conditioned subtraction detects fewer injected faults than the plain source-side error it is built from. In the realized test split, both principal detectors incur the same 0.67% false-alarm rate while subtraction loses 7.27 detection points. The nested bootstrap explains why the original, narrower interval should not be the headline number: once the calibration sample and every learned threshold are resampled, the detection-difference interval widens from roughly 0.8 points to roughly 5.0 points. Even so, the widened interval remains entirely negative, from -8.80 to -3.81 points. For a team deciding whether to add a rotated-input check to an existing differential test suite, this is the single most load-bearing number in the paper: even under the most pessimistic, calibration-uncertainty-inflated view of the evidence, subtraction still loses.

The nested false-alarm-rate distribution supplies an equally important qualification. The observed false-

Table 1: Primary detector comparison on the held-out CIFAR-100 test split (16,200 fault cases, 900 clean cases, thresholds calibrated on a disjoint 300-image sample per model).

Detector	Fault detection rate (TPR)	Clean false-alarm rate (FPR)	AUROC	AUPRC
Metamorphic contrast (subtracted)	75.48%	0.67%	0.921	0.995
Direct L2 (plain comparison)	82.75%	0.67%	0.954	0.997
Mixed absolute/relative tolerance	67.10%	1.22%	0.878	0.993

Takeaway: At an identical 0.67% false-alarm rate, subtracting the two comparisons (metamorphic contrast) catches noticeably fewer injected faults than simply keeping the plain, single-input comparison (direct L2); the ad hoc mixed-tolerance rule trails both and has a higher false-alarm rate.

Table 2: Execution-cost-matched comparison: giving the plain comparison the same two executions used by metamorphic contrast, without subtracting them.

Method	Inputs used	Detection rate (TPR)	False-alarm rate (FPR)	Top-1 localization accuracy
Metamorphic contrast (subtracted)	source + rotated	75.48%	0.67%	15.15%
Two-input direct maximum (not subtracted)	source + rotated	84.52%	1.11%	45.19%

Takeaway: Combining the same two executions by taking their maximum, instead of subtracting them, outperforms subtraction on both detection and localization – the extra input was never the problem; discarding the shared signal by subtracting was.

alarm-rate difference in the primary split is exactly zero, but repeated calibration resampling produces an interval from -1.00 to +1.11 points. Independently calibrated 99th-percentile thresholds, estimated from only 300 clean images, do not guarantee identical realized false-alarm rates in general. The right conclusion is not that the two detectors have intrinsically equal false-alarm behavior, but that direct L2's detection advantage holds at the same realized false-alarm rate in the primary split and stays negative across the whole nested resampling distribution, while its false-alarm-rate difference from metamorphic contrast remains unresolved.

The two-input direct maximum sharpens the compute-budget interpretation without fully resolving operating-point fairness. Metamorphic contrast consumes both a source and a transformed execution, while the plain source-only detector uses only the first. The maximum keeps both direct error magnitudes and reaches 84.52% detection at 1.11% false-alarm rate, showing the transformed execution genuinely contains useful direct evidence, and that using two inputs does not require subtracting them. It does not, on its own, prove superiority at an exactly matched false-alarm rate, since metamorphic contrast realizes a lower 0.67% rate; a future confirmatory comparison should pre-specify an exact-rate interpolation.

The error-geometry analysis explains why this outcome is coherent rather than surprising. Subtraction is, mechanically, a directional filter. When a fault changes outputs in a similar way on the source and the rotated input – plausible for a weight perturbation, since the same modified parameter is used in both executions – the two error vectors are positively aligned, and their shared component is exactly what gets removed by subtracting one from the other. The observation that direct-only detections are more aligned than metamorphic-only detections matches this description precisely.

Table 3: Layer-localization accuracy across all 16,200 fault cases: naming the perturbed convolution from clean-normalized activation scores.

Scoring method	Top-1 accuracy	Top-3 accuracy	Mean reciprocal rank
Metamorphic activation contrast (subtracted)	15.15%	29.40%	0.288
Direct activation scoring (source only)	45.15%	56.79%	0.540

Takeaway: Subtraction is even costlier for localization than for detection: the plain activation score names the perturbed layer as its single top guess roughly three times as often.

Table 4: Sensitivity of the metamorphic-versus-direct detection gap to the rotation angle and interpolation method (150/150 calibration/test images per condition; primary condition uses 300/300).

Rotation condition	Metamorphic TPR	Direct TPR	Difference	Metamorphic FPR
Primary: 10°, bilinear	75.48%	82.75%	-7.27 pts	0.67%
5°, bilinear	74.41%	83.69%	-9.28 pts	1.11%
20°, bilinear	80.12%	83.69%	-3.57 pts	2.44%
10°, nearest-neighbor	77.47%	83.69%	-6.22 pts	0.89%
10°, bicubic	74.75%	83.69%	-8.94 pts	1.33%

Takeaway: No rotation angle or interpolation method tested reverses the ordering; the gap narrows somewhat at a larger 20° rotation, but that comes with a higher metamorphic false-alarm rate and a less semantically stable clean transformation.

That match is not, by itself, causal validation. The cosine of the angle between the two errors and the size of the subtracted contrast share the same underlying vectors, so a strong correlation between them is expected from the identity itself rather than an independent finding. No variable here was independently manipulated, and no alignment-based rule was evaluated on held-out fault families or transformations excluded from its own formulation. Within this paper, alignment is only a post-primary consistency diagnostic; treating it as a genuine predictor would require a separate protocol that fits a rule without access to each evaluation fault, then measures its predictive accuracy prospectively – exactly the kind of study the reproduced historical-defect corpus is positioned to support next.

The rotation-sensitivity grid points to a practical tension worth naming directly. A stronger 20-degree rotation narrows the detection deficit, but it also lowers the reference model's own prediction stability to 52% and raises the metamorphic detector's false-alarm rate to 2.44%. A transformation strong enough to decorrelate a fault's errors on the two inputs can also destabilize the model's ordinary, fault-free behavior; a transformation gentle enough to leave clean behavior stable may simply not decorrelate the fault enough to help. A practitioner tempted to simply crank up the rotation angle to close the gap observed here should read this as a warning, not a recipe: the fix trades one problem for another.

For anyone building this kind of hybrid detector in practice, the design lesson is to preserve components rather than store only their contrast. Whenever executions are available for both x and a transformed T(x), it costs little extra to record the source-error size, the transformed-error size, and the subtracted-contrast size separately, calibrate each on clean data independently, and only then decide, with evidence, which combination rule to deploy. Discarding the direct magnitudes at collection time makes this later comparison impossible. Execution budget deserves the same discipline: any detector that uses a follow-up input should be compared against a transparent combination of the direct errors it already has access to, such as the maximum used here, before being credited with an improvement that might really be coming from the extra execution

alone.

The localization result carries a similar warning, in sharper form. A transformed activation contrast can look diagnostically appealing because it asks where a relation changes, rather than just where an output changes. Here, however, direct activation discrepancies persist across both inputs often enough that subtracting them away costs more than it gains: the cost-matched activation maximum performs almost identically to the plain, single-input activation score, suggesting the transformed input adds little extra localization information once a maximum, rather than a difference, combines it. This conclusion is bounded to the compact activation sketches used here; full tensors, gradient-based attribution, or pass-level analysis may behave differently [4,8,19].

None of this argues against metamorphic testing where no second implementation exists to compare against at all – in a single-implementation setting, a source/follow-up relation may be the only executable oracle available, and this study says nothing about that case. Nor does it diminish transformations as test generators: DeepHunter, MT-DLComp, EAGLE, and model-level metamorphic testing all use transformations to reach behavior a source-only test might never exercise [5,10,11,17], a goal this study neither tests nor contradicts. This paper's result concerns something narrower: the hybrid setting in which candidate-minus-reference discrepancies on two related inputs have already been computed, and the only remaining question is how to combine them.

The fault-level heterogeneity in the results also indicates where future evidence should concentrate effort. High-severity fault conditions saturate every detector and therefore cannot distinguish between scoring rules; the informative cases sit near the calibrated detection boundary, especially low-severity noise and scaling. Future evaluation matrices should deliberately span a range of effect sizes, avoid conditions that trivially saturate or fail for every method, and hold out entire fault families or relations for genuine, prospective evaluation rather than testing on the same faults used to design the scoring rule.

The historical issue corpus reproduced here provides raw material for that next study, but not the study itself. Turning it into one requires a semantically justified source/follow-up relation for each real defect, clean controls sufficient to calibrate a low false-alarm-rate threshold, and localization ground truth derived from the actual maintainer patch rather than from a convenient proxy. A leave-one-defect-out protocol could then test directly whether an alignment-derived rule, fit without access to a given defect, predicts which scoring method will be more sensitive to it at a matched false-alarm rate. Until that experiment exists, this paper supports only a descriptive cancellation account measured on injected, controlled drift.

For practitioners weighing whether to add a metamorphic contrast to an existing differential test, the conservative recommendation is to treat the choice of relation and the choice of scoring rule as two separate decisions. Validate the semantic content of the transformation on its own terms; keep the direct error components rather than discarding them; account for calibration uncertainty rather than trusting a single threshold estimate; report each detector's own realized false-alarm rate rather than assuming a nominal one; compare execution budgets honestly; and require a matched operating point before claiming one method is stronger than another. A relation can be semantically sound while the numerical rule built on top of it is nonetheless counterproductive – and the two need to be checked independently, not assumed to rise or fall together.

6 Limitations

Construct validity of the faults. The primary matrix uses deterministic weight scaling, additive Gaussian noise, and symmetric quantization rather than a probability sample of historical runtime, compiler, converter, or kernel defects. These controlled perturbations give exact location labels and known severity, but they may also produce especially persistent cross-input errors compared with real-world bugs. Input-selective optimizations, discontinuities, exceptional values, nondeterministic kernels, control-flow changes, and shape

specialization could all produce different error geometry than the weight perturbations studied here; the detection rates reported estimate sensitivity to this specific fault matrix, not recall over the population of deployed failures such as the ones sketched in the opening scenarios.

Construct validity of the relation. The 10-degree rotation is assumed to preserve the CIFAR-100 class label, and no human annotation verifies individual pairs. Mean reference-model prediction stability across the rotation is 71.39% in the primary condition and as low as 52.00% at 20 degrees. Prediction stability is neither necessary nor sufficient for a rotation being semantically valid on a given image; the stable/unstable subgroup results show that prediction changes do not fully explain the detector ordering, but they cannot, by themselves, validate the relation's semantics.

Internal validity of the runtime contrast. The runtime version and the optimization flag change together in the primary comparison. A separate factor-isolation analysis records zero clean-input effect from the version alone at fixed optimization, and identical small discrepancies from enabling optimization in either version – but the full 54-fault matrix was not repeated across all four factorial cells, so the primary estimand remains the combined-configuration contrast, not a causal claim about the version upgrade alone.

Protocol provenance. The primary protocol was recorded locally in the project's own version history rather than externally registered or independently timestamped, and the protocol and the primary results first appear together in that history, so it cannot independently establish that the protocol was frozen strictly before execution. It constrains and documents the intended analysis, but does not carry the same evidentiary weight as a verifiably preregistered study.

Calibration and statistical uncertainty. Only 300 clean observations per model estimate each detector's 99th-percentile threshold. The original held-out-image bootstrap held these thresholds fixed and so omitted calibration uncertainty entirely, which is why its very narrow interval should not be treated as the headline uncertainty estimate. The added nested bootstrap corrects this omission, by resampling calibration images, re-estimating thresholds, and resampling held-out images together – but it still conditions on the observed pools, the three selected models, the 54 fault configurations, and the rotation and runtime pair studied here; it does not sample new architectures, defects, transformations, or hardware, and its percentile distribution is not a population-general confidence statement about machine-learning deployments in general.

Operating-point validity. Independently calibrated nominal 99th-percentile thresholds do not guarantee equal realized false-alarm rates. Direct and metamorphic scores happen to match exactly at 0.67% in the primary split, while their nested false-alarm-rate-difference interval spans zero broadly. The cost-matched maximum realizes a different 1.11% false-alarm rate, so its 84.52% detection rate cannot be read as an exactly matched-rate comparison against metamorphic contrast's 75.48% at 0.67%. No pre-specified receiver-operating-curve interpolation or cross-validated common-rate threshold exists in this study, so the compute-budget comparison remains exploratory.

Localization validity. Ground truth for localization is the convolution whose weights were directly modified. The originating weight tensor, the first layer where a symptom becomes visible, the responsible compiler pass, and the most actionable place to look for a repair can all be different things in a real system. Localization here uses per-channel means, standard deviations, and maxima compressed to at most 64 features per layer; this compression may erase spatially localized errors, and the results do not generalize to full tensors, gradient-based attribution, or compiler-pass-level localization. The localization intervals also use fixed clean normalization thresholds that the nested revision does not re-estimate.

Mechanism validity. Cross-input error cosine and the cancellation ratio are algebraically coupled through the same two error vectors, and detector membership is itself derived from those vectors, so their correlation and the discordant-group difference are expected consistency checks, not independent causal validation. Alignment has not been evaluated as a genuine out-of-sample predictor on faults, relations, or historical defects excluded from its own formulation, and this paper makes no predictive-performance claim on that basis.

Status of post-primary analyses. The cost-matched detector, the rotation-sensitivity grid, the runtime-

factor isolation, the error-geometry analysis, and the nested bootstrap were all formulated after observing the primary result or reviewing its initial uncertainty. Recording each of these procedures before running its own computation reduces flexibility within that individual analysis, but it does not turn any of them into a confirmatory test of the original hypothesis; they explain, stress-test, and help interpret the primary result rather than altering the primary conjunctive decision itself.

External validity. The detector experiment covers three related CIFAR-100 convolutional networks, CPU execution, one primary rotation relation, and one ONNX Runtime version pair. It excludes transformer architectures, language and audio models, regression tasks, GPU and other accelerator execution, stochastic inference, dynamic input shapes, real production traffic, and other runtime families. The three models function as replications within this matrix rather than a random sample from the space of possible architectures. The manufacturing and radiology scenarios used to motivate the study are illustrative framings for why silent drift matters, not settings that were themselves measured.

Historical-corpus validity. The 14 attempted public issues form a convenience-constrained set, selected specifically because an executable oracle and a compatible environment could be assembled for them. Ten reproduced, four did not, and one further configuration was hardware-incompatible; these outcomes demonstrate that the reproduction effort recovered genuine defects, not that the corpus is representative of runtime or framework reliability broadly, and it lacks the shared clean controls, matched-false-alarm-rate scoring, and maintainer-grounded localization labels that a defect-level detector-effectiveness claim would require.

Reproducibility validity. The exact-replication check reruns the same machine, environment, model files, sample manifest, runtime installations, and random seeds. Byte-for-byte equality between the two runs demonstrates deterministic replay and guards against accidental drift in the analysis code; it is not an independent laboratory replication, a fresh draw of data, or evidence about numerical identity across different hardware.

7 Reproducibility

All numerical claims are linked to locally recorded protocols, manifests, raw tables, summaries, and checksums held in the accompanying artifact package. The primary workflow uses a checksum-bound CIFAR-100 revision, three checksum-pinned ONNX models, a deterministic disjoint 300/300 sample split, a 10-degree bilinear rotation, two pinned ONNX Runtime configurations, 54 deterministic fault configurations, and recorded detector and localization formulas.

From the artifact root, a bootstrap script reconstructs the study environments when dependencies are not provisioned, a rerun script regenerates the primary workflow and same-environment replay, and a separate script refreshes the original manuscript analyses. The post-primary analyses (the cost-matched baseline and the rotation/interpolation follow-up conditions) are regenerated with dedicated analysis scripts, and the nested uncertainty analysis is regenerated with a script that accepts a replicate count (5000 in the reported run). It reads cached primary executions, resamples 300 calibration source images, recomputes all per-model primary detector thresholds, independently resamples 300 held-out source images, and preserves model/fault/detector pairing, writing a machine-readable distribution and summary whose checksums are recorded in the artifact manifest.

A claim-checking script verifies the original headline values against recorded JSON evidence, and a checksum manifest verifies all original manuscript artifacts. A focused automated test suite exercises the analysis code. A separate script regenerates the deduplicated historical-issue summary. These checks establish implementation behavior and artifact integrity, not scientific effectiveness.

The exact replay reproduces raw detection scores, localization ranks, the fault manifest, and the sample manifest byte-for-byte over identical frozen inputs. Network access is required only to reacquire upstream

assets or reconstruct package environments; recorded evidence and integrity checks are inspectable offline. Model files, the CIFAR-100 data revision, and every generated artifact are checksummed so results can be verified without rerunning the full pipeline.

8 Conclusion

Return to the factory floor and the radiology queue from the opening of this paper: a team upgrades its inference runtime for speed, nothing crashes, and a rare class of cases quietly starts behaving differently – a scratched part that passes inspection, or a borderline scan that no longer sits above the flagging threshold. The instinct to build a smarter test for exactly this situation – one that compares a model's behavior on an image and on a lightly rotated copy of it, and looks at what changes between the two – is reasonable, and it is the instinct a careful engineer should have. This paper's evidence says that instinct needs a caveat before it is trusted in production. On three public CIFAR-100 classifiers, under two pinned ONNX Runtime configurations, and across 54 controlled weight-perturbation faults, subtracting the two comparisons caught fewer injected faults than simply keeping the comparison on the original image alone – 75.48% versus 82.75% detection at the identical 0.67% false-alarm rate – and the deficit survived a bootstrap that redid the calibration itself from scratch five thousand times (95% interval -8.80 to -3.81 points). Pinpointing which layer had actually been perturbed showed the same pattern even more sharply, 15.15% versus 45.15%. Giving the plain comparison the same two executions, without subtracting them, closed the gap and then some, reaching 84.52% detection – so the extra input was never the problem; the subtraction was.

The reason is not exotic, and it is worth restating in plain terms one last time because it is the single idea this whole paper turns on: when a fault behaves persistently rather than input-specifically – the same corrupted weight doing the same thing regardless of which photograph passes through it – it nudges both the original and the rotated comparison in roughly the same direction. Subtracting one from the other throws away exactly that shared signal, keeping only the noise left over. A test built to be cleverer than a simple comparison ended up being worse than one, not because the underlying idea of comparing a photograph to a rotated copy of itself was wrong, but because of what happened to the arithmetic built on top of a perfectly reasonable idea.

None of this is a verdict against metamorphic testing as a strategy – it says nothing about settings where no second implementation exists to compare against, or about transformations used purely to reach new behavior rather than to build a subtracted contrast. It is a bounded result about one specific, common-looking design choice, measured honestly against the plain baseline it was meant to improve on. It is not the last word on whether the same subtraction would behave the same way against a real historical bug, a different transformation, or a different kind of model, and this paper has tried to be explicit about exactly where that boundary sits.

The takeaway worth carrying past this paper, for a reader who forgets everything else in it, is correspondingly narrow, concrete, and – the authors hope – memorable: an extra, related input makes a test more expensive automatically, but it does not make a test more sensitive automatically. That has to be checked, one baseline at a time, at the same false-alarm rate, before anyone trusts it on a factory floor, in a hospital, or anywhere else a quiet numerical drift is allowed to hide behind an unchanged accuracy number.

References

[1] Sergio Segura, Gordon Fraser, Ana B. Sánchez, Antonio Ruiz-Cortés. A Survey on Metamorphic Testing. IEEE Transactions on Software Engineering. 2016. doi: 10.1109/TSE.2016.2532875. https://eprints.whiterose.ac.uk/id/eprint/110335/

[2] Junhua Ding, Xiaojun Kang, Xin-Hua Hu. Validating a Deep Learning Framework by Metamorphic Testing. IEEE/ACM International Workshop on Metamorphic Testing. 2017. doi: 10.1109/MET.2017.2. https://www.proceedings.com/content/035/035175webtoc.pdf

[3] Kexin Pei, Yinzhi Cao, Junfeng Yang, Suman Jana. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. ACM Symposium on Operating Systems Principles. 2017. doi: 10.1145/3132747.3132785. https://www.cs.columbia.edu/~junfeng/papers/deepxplore-sosp17.pdf

[4] Hung Viet Pham, Thibaud Lutellier, Weizhen Qi, Lin Tan. CRADLE: Cross-Backend Validation to Detect and Localize Bugs in Deep Learning Libraries. International Conference on Software Engineering. 2019. doi: 10.1109/ICSE.2019.00107. https://www.cs.purdue.edu/homes/lintan/publications/cradle-icse19.pdf

[5] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, Simon See. DeepHunter: A Coverage-Guided Fuzz Testing Framework for Deep Neural Networks. International Symposium on Software Testing and Analysis. 2019. doi: 10.1145/3293882.3330579. https://experts.illinois.edu/en/publications/deephunter-a-coverage-guided-fuzz-testing-framework-for-deep-neur/

[6] Xiaofei Xie, Lei Ma, Haijun Wang, Yuekang Li, Yang Liu, Xiaohong Li. DiffChaser: Detecting Disagreements for Deep Neural Networks. International Joint Conference on Artificial Intelligence. 2019. doi: 10.24963/ijcai.2019/800. https://www.ijcai.org/proceedings/2019/800

[7] Zan Wang, Ming Yan, Junjie Chen, Shuang Liu, Dongdi Zhang. Deep Learning Library Testing via Effective Model Generation. ESEC/FSE. 2020. doi: 10.1145/3368089.3409761. https://tjusail.github.io/projects/files/lemon.pdf

[8] Qianyu Guo, Xiaofei Xie, Yi Li, Xiaoyu Zhang, Yang Liu, Xiaohong Li, Chao Shen. Audee: Automated Testing for Deep Learning Frameworks. Automated Software Engineering. 2020. doi: 10.1145/3324884.3416571. https://personal.ntu.edu.sg/yi_li/files/Guo2020AAT.pdf

[9] Ming Yan, Junjie Chen, Xiangyu Zhang, Lin Tan, Gan Wang, Zan Wang. Exposing Numerical Bugs in Deep Learning via Gradient Back-Propagation. ESEC/FSE. 2021. doi: 10.1145/3468264.3468612. https://www.cs.purdue.edu/homes/lintan/publications/grist-icse21.pdf

[10] Dongwei Xiao, Zhibo Liu, Yuanyuan Yuan, Qi Pang, Shuai Wang. Metamorphic Testing of Deep Learning Compilers. Proceedings of the ACM on Measurement and Analysis of Computing Systems. 2022. doi: 10.1145/3508035. https://researchportal.hkust.edu.hk/en/publications/metamorphic-testing-of-deep-learning-compilers-2/

[11] Jiannan Wang, Thibaud Lutellier, Shangshu Qian, Hung Viet Pham, Lin Tan. EAGLE: Creating Equivalent Graphs to Test Deep Learning Libraries. International Conference on Software Engineering. 2022. doi: 10.1145/3510003.3510165. https://jiannanwang.github.io/files/eagle-icse22.pdf

[12] Jiazhen Gu, Xuchuan Luo, Yangfan Zhou, Xin Wang. Muffin: Testing Deep Learning Libraries via Neural Architecture Fuzzing. International Conference on Software Engineering. 2022. doi: 10.1145/3510003.3510092. https://arxiv.org/abs/2204.08734

[13] Jiawei Liu, Yuxiang Wei, Sen Yang, Yinlin Deng, Lingming Zhang. Coverage-Guided Tensor Compiler Fuzzing with Joint IR-Pass Mutation. Proceedings of the ACM on Programming Languages, OOPSLA1. 2022. doi: 10.1145/3527317. https://lingming.cs.illinois.edu/publications/oopsla2022.pdf

[14] Gan Wang, Zan Wang, Junjie Chen, Xiang Chen, Ming Yan. An Empirical Study on Numerical Bugs in Deep Learning Programs. Automated Software Engineering NIER Track. 2022. doi: 10.1145/3551349.3559561. https://conf.researchr.org/details/ase-2022/ase-2022-nier-track/18/An-Empirical-Study-on-Numerical-Bugs-in-Deep-Learning-Programs

[15] Jiawei Liu, Jinkun Lin, Fabian Ruffy, Cheng Tan, Jinyang Li, Aurojit Panda, Lingming Zhang. NNSmith: Generating Diverse and Valid Test Cases for Deep Learning Compilers. ASPLOS. 2023. doi: 10.1145/3575693.3575707. https://arxiv.org/abs/2207.13066

[16] Purvish Jajal, Wenxin Jiang, Arav Tewari, Erik Kocinare, Joseph Woo, Anusha Sarraf, Yung-Hsiang Lu, George K. Thiruvathukal, James C. Davis. Interoperability in Deep Learning: A User Survey and Failure Analysis of ONNX Model Converters. International Symposium on Software Testing and Analysis. 2024. doi: 10.1145/3650212.3680374. https://ecommons.luc.edu/cs_facpubs/349/

[17] Yanzhou Mu, Juan Zhai, Chunrong Fang, Xiang Chen, Zhixiang Cao, Peiran Yang, Kexin Zhao, An Guo, Zhenyu Chen. Improving Deep Learning Framework Testing with Model-Level Metamorphic Testing. Proceedings of the ACM on Software Engineering, ISSTA. 2025. doi: 10.1145/3728972. https://people.cs.umass.edu/~juanzhai/papers/issta25.pdf

[18] Kshitij Dubey, Benjamin Driscoll, Anjiang Wei, Neeraj Kayal, Rahul Sharma, Alex Aiken. Equivalence Checking of ML GPU Kernels. arXiv preprint. 2025. doi: 10.48550/arXiv.2511.12638. https://arxiv.org/abs/2511.12638

[19] Nikolaos Louloudakis, Ajitha Rajan. DiTOX: Fault Detection and Localization in the ONNX Optimizer. ACM SIGPLAN International Conference on Compiler Construction. 2026. doi: 10.1145/3771775.3786265. https://github.com/luludak/DiTOX