OBSERVATION LIMITS OF NODE ATTRIBUTION UNDER GRAPH-LEVEL SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Node attribution ranks nodes by their relevance to a graph-level outcome, supporting applications such as localizing faulty program statements from test outcomes. Existing evaluations emphasize faithfulness to a trained model, leaving unresolved whether the observations distinguish the responsible node from other candidates. We study the limits that available observations impose on node attribution in real software faults. Our analysis characterizes attainable localization accuracy, relates indistinguishability under shared message passing to equal node scores, and gives upper confidence bounds from observation counts collected separately for each candidate target. To assess the prevalence of this ambiguity, we enrich test coverage with local program descriptors and identify repaired and non-repaired statements that remain indistinguishable under these observations. Such pairs occur in 34.8% of eligible Defects4J faults and 33.1% of eligible Codeflaws faults. The three evaluated architectures and three coverage-based scoring formulas tie on every matched pair, yielding pair AUC 0.50. We then separate the effects of target information and training supervision by independently varying node labels and a node feature encoding oracle target information. On structurally interchangeable EPFL circuit nodes, this target anchor alone gives mean AUC 0.980, compared with 0.511 for node supervision alone, while combining both gives mean pair AUC 1.0. These results connect ambiguity in real workloads to the information required for localization and motivate evaluating node attribution separately from graph-level prediction.

1 INTRODUCTION

Node attribution ranks nodes by their relevance to a graph-level outcome. Under graph-level supervision, training labels describe whole graphs, while the desired explanation concerns individual nodes. Fault localization provides a concrete example: tests reveal program behaviour, and a localizer ranks statements that may be responsible for a failure. Graph neural networks can use coverage and program structure to learn these rankings, as in GRACE (Lou et al., 2021).

A related line of work explains the predictions of a trained graph model. GNNExplainer and PGExplainer identify influential subgraphs or feature masks (Ying et al., 2019; Luo et al., 2020), while SubgraphX and GNN-LRP attribute predictions through subgraphs and walks (Yuan et al., 2021; Schnake et al., 2022). Explanation benchmarks and analyses assess agreement with reference targets, model faithfulness and stability (Agarwal et al., 2023; 2022). Faithfulness describes agreement with the trained predictor. Localizing an externally specified target, such as a statement identified by a repair, additionally requires observations that distinguish that target from competing locations. Accurate prediction alone can coexist with explanations unrelated to the model's underlying computations (Azzolin et al., 2026).

For example, a repaired statement and a non-repaired statement can have identical test coverage, a recognized ambiguity in classical fault localization (Baudry et al., 2006). Adding syntax and control-flow descriptors can separate some such candidates. The remaining pairs raise two questions: how often do the supplied observations still fail to distinguish the target, and what additional information resolves that ambiguity? These questions concern the observations available for localization as well as the model used to process them.

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.

We therefore analyze information limits and representation limits separately. The Weisfeiler–Leman (WL) test refines node neighbourhood descriptions, providing a standard characterization of distinctions preserved by shared message passing (Weisfeiler & Leman, 1968; Xu et al., 2019; Morris et al., 2019). Higher-order and subgraph representations can preserve additional distinctions (Maron et al., 2019; Bevilacqua et al., 2022), and encoder-relative identifiability studies the information retained by a chosen representation (Yan et al., 2025). Our statistical analysis instead starts from all observations supplied to the learner and asks whether they distinguish alternative target locations.

We call two targets *observation equivalent* when they induce the same distribution of these observations. Equally likely equivalent targets have chance-level optimal localization accuracy, regardless of the estimator. Separately, for shared message-passing scorers, nodes with the same refined neighbourhood representation receive identical scores. This model-level result predicts pairwise ties under specified structural conditions (Section 4). When the target-conditioned observation laws are unknown, independent observation counts collected separately for each target yield upper confidence bounds on attainable localization accuracy (Proposition 4).

We evaluate these distinctions in real faults and controlled interventions. After enriching test coverage with local program descriptors, matched repaired and non-repaired statements remain in 34.8% of eligible Defects4J faults and 33.1% of eligible Codeflaws faults (Just et al., 2014; Tan et al., 2017). The evaluated shared-head architectures and coverage-based scoring formulas tie on these matched pairs, connecting the representation analysis to the observed ranking behaviour (Section 5.2). In executable circuit experiments, an oracle target anchor adds target information as a node feature. Varying this feature independently of node supervision distinguishes information supplied at prediction time from labels supplied during training (Section 5.3 and Section 5.4.1).

Our contributions are:

1. We formalize observation equivalence for node targets and characterize the localization accuracy supported by their observation laws (Theorem 1).
2. We establish score equality for the specified shared message-passing scorers and derive finite-sample upper confidence bounds from per-target observation counts (Lemma 2 and Proposition 4).
3. We measure unresolved candidate pairs in real software faults and use controlled target-information interventions to separate the roles of observations, representations and node supervision.

2 Related Work

Graph expressivity. Graph expressivity studies which structural distinctions an architecture can represent through the Weisfeiler–Leman hierarchy (Weisfeiler & Leman, 1968). Message-passing characterizations connect neighbourhood aggregation to this hierarchy (Morris et al., 2019; Xu et al., 2019). Higher-order representations and subgraph aggregation extend discrimination by representing node tuples or combining extracted subgraphs (Maron et al., 2019; Bevilacqua et al., 2022). Structural discrimination leaves the target unspecified when multiple candidate locations remain compatible with the observed graph and label. We group targets by WL colour and bound localization from complete observations independently of the architecture.

Node identifiability and explanation reliability. Yan et al. (2025) bound source-node recovery for encoders on random regular graphs and lift it with context anchors carrying shortest-path distances. Our circuit comparison isolates the same lift with a one-coordinate anchor and separates it from the supervision used to train the scorer. Explanation taxonomies organize accounts of trained model behaviour (Yuan et al., 2023). Analyses characterize faithfulness and stability (Agarwal et al., 2022; Tai et al., 2025; 2026) alongside causal necessity and sufficiency (Cai et al., 2022), with formal semantics for self-explaining models (Azzolin et al., 2025) and connections between gradient and perturbation attribution (He et al., 2025). GraphXAI scores explanations against ground-truth masks supplied with its benchmark graphs and against model behaviour (Agarwal et al., 2023), while WL-aligned benchmarks construct targets for scoring explanations (Fontanesi et al., 2026). Such a target accompanies every input by construction. A program repair instead fixes an external target that the observations may not identify. Azzolin et al. (2026) construct explanations that preserve a good prediction while carrying no semantics, and our matched pairs are the complementary case in which

no explanation can carry the target. Our finite-sample bound complements them by quantifying the observation information available to identify an external node target.

Weak supervision and partial identification. Multiple-instance learning predicts bag labels without training labels for every constituent instance (Dietterich et al., 1997), including attention-based aggregation (Ilse et al., 2018) and graph representations of instance relationships (Tu et al., 2019). Programmatic weak supervision combines noisy labeling functions through a label model (Ratner et al., 2016). In our setting, alternative node assignments are observation-equivalent when they induce the same law of the supplied graph observations. Predicting an aggregate label can leave the relevant member unidentified. We quantify attainable localization from sampled observation laws for targets within WL colour classes.

Fault localization and digital test. Coverage-based fault localization groups statements executed by the same tests into indistinguishable dynamic basic blocks (Baudry et al., 2006). Analyses examine ambiguity in locator evaluation (Steimann et al., 2013) and spectrum formulas (Xie et al., 2013), while test prioritization reduces ambiguity groups (Gonzalez-Sanchez et al., 2011). Digital-test research characterizes fault equivalence (McCluskey & Clegg, 1971) and diagnostic resolution (Waicukauski & Lindbloom, 1989), including indistinguishable path-delay faults (Tekumalla & Davidson, 2002). These results describe which locations a specified program or circuit and its tests can distinguish. Our study measures how often matches survive descriptors beyond coverage and connects the surviving pairs to observation-based localization bounds under graph-level supervision.

Classification bounds and localization certificates. Classification-error analysis characterizes the best accuracy supported by class distributions independently of a learned classifier's capacity (Berisha et al., 2016; Theisen et al., 2021). Berisha et al. (2016) bound the Bayes error of two classes by a divergence estimated from samples, which fixes the class-conditional laws in advance. For specified generative families, Bayes-error calculations provide a prediction reference (Theisen et al., 2021). The corresponding localization problem requires specifying which candidate node state generates each observation law. We express that problem through conditional observation counts and incorporate their sampling uncertainty in an upper confidence bound on attainable target identification. Appendix B compares the target definitions and mechanisms of these research lines.

3 SETUP

We first specify the visible graph information and hidden node targets. We then define target assignments that preserve the observations, preparing separate analyses of statistical identification and representation-based ranking.

3.1 OBSERVATIONS AND RANKING MEASURES

We specify what the learner observes before defining the hidden target.

Let $G = (V, E, X)$ be a graph with node set V, edge set E and node types $X : V \to \Sigma_0$ in a finite alphabet. Let $N(v)$ denote the neighbours of node v. Training supplies graph labels $Y(G)$. The hidden relevance target is $\ell : V \to \{0, 1\}$. In the fault-localization instantiation, nodes represent statements and tests, and edges encode test coverage. Statement features combine coverage-derived spectrum statistics with local program descriptors. Test outcomes provide the training labels and also enter those spectrum statistics. A repair supplies the statement-level evaluation target, withheld from inputs and ordinary training. Appendix I specifies the descriptor encoding and Appendix H the test-outcome objective. A scorer $s(G, v)$, abbreviated $s(v)$, assigns larger values to greater relevance.

Nodes that share a representation. The one-dimensional Weisfeiler–Leman test (1-WL) describes the distinctions preserved by shared message passing (Weisfeiler & Leman, 1968; Xu et al., 2019; Morris et al., 2019). Starting from the node types, each iteration refines colours using the previous colour and the multiset of neighbours' colours. Fix refinement depth k, corresponding to the k message-passing layers considered below. Write $c(v) = c_G^{(k)}(v)$ and let C denote a colour class. Colours index refined descriptions rather than additional input features (Definition 5). Equal colours mean equal descriptions at that depth. For example, two statement nodes with the same attributes and the same test neighbours receive the same messages. Adding local syntax can distinguish some such

nodes by changing their initial attributes. Sharing attributes and neighbours is a sufficient condition for equal colours. Finite-depth colour equality also covers nodes with different neighbourhoods that refinement has not separated.

A feature-preserving automorphism permutes nodes while preserving edges and attributes. An *automorphic witness pair* (w_1, w_2) has exactly one relevant member and admits an automorphism preserving all observed attributes that exchanges the two nodes. Such a pair shares a colour at every iteration. The shared encoder, message maps and scoring head in Lemma 2 then yield identical scores.

A scorer is *$\leq$1-WL representable at depth k* when equal iteration-k colours imply equal scores within every fixed graph.

Ranking performance. Overall ranking can be accurate across representation classes while leaving an ambiguous pair unresolved. We therefore use both tie-aware attribution AUC and a local pair AUC. For a specified finite node collection containing both relevance classes, attribution AUC averages all relevant–irrelevant comparisons, assigning ties weight $\frac{1}{2}$. The theory permits a single graph or the disjoint union of a fixed graph collection, including cross-graph comparisons in the latter case. Pair AUC averages only the specified one-relevant-member witness pairs with the same tie weight. It measures pairwise ranking, separately from top-1 target identification. Experimental pooling units are stated with each result.

3.2 Candidate relevance assignments

To isolate ambiguity in the target, we compare assignments that preserve the observed graph and its label. Over a finite graph collection $\mathcal{G}$, compare *worlds* ℓ^A and ℓ^B that assign binary node relevance under the following conditions.

Assumption 1 (Decoupled label, W1)**.** The graph label $Y(G)$ is a function of the observable graph alone and is identical across worlds. The observable data $(G, Y(G))_{G\in\mathcal{G}}$ is world-independent.

Assumption 2 (Within-colour swap, W2)**.** For every graph G and every colour class C, the relevant-node counts agree, so $\sum_{v\in C} \ell^A(v) = \sum_{v\in C} \ell^B(v)$.

Assumption 3 (Non-triviality, W3)**.** Some node has $\ell^A(v) \neq \ell^B(v)$.

For example, world A can mark w_1 relevant and world B can mark w_2 relevant while all other labels remain unchanged, as in Figure 1. W1 fixes the learner's input and training labels. W2 fixes how much relevance each representation class contains while allowing its members to exchange relevance. W3 ensures that this exchange changes the target. W1 supports learning invariance, while W2 preserves the positive and negative counts needed for the AUC identity. W3 makes the two targets distinct. These conditions describe a fixed graph collection, before introducing a probability law for choosing a target. In the controlled graphs we construct the alternative assignments explicitly. In real faults we measure representation matching and score equality for repaired/non-repaired statements. The statistical analysis of target-conditioned observation laws is evaluated separately using controlled channels.

4 The Barrier

We derive three complementary results: a statistical limit on target-identification accuracy, a ranking-invariance result for shared representations, and a finite-sample upper confidence bound on optimal accuracy from target-conditioned counts.

4.1 Identification from the complete observations

For the statistical analysis, we select one target Z from m specified candidate locations, in place of a full binary assignment ℓ. A witness pair gives the case $m = 2$. Let O include all training and evaluation information allowed to the localizer. The *observation channel* comprises the target-conditioned laws of O. Its *Bayes success probability* maximizes $\Pr\{\widehat{Z}(O) = Z\}$ over estimators. For distributions P, Q, total variation $\mathrm{TV}(P, Q) = \sup_D |P(D) - Q(D)|$ measures their largest probability difference over measurable events D.

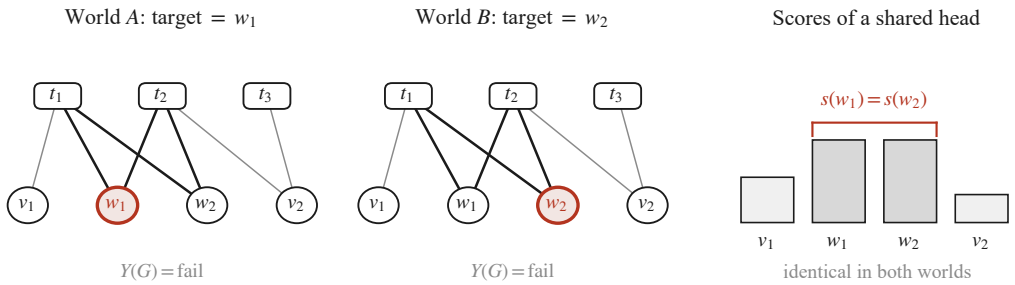

Figure 1: **A controlled pair of relevance assignments satisfying W1–W3.** The two worlds preserve the graph, descriptors and graph label while exchanging which matched statement is relevant. Equal attributes and test neighbourhoods give equal representations under Lemma 2, hence equal scores and pair AUC $\frac{1}{2}$.

The following theorem treats exact equivalence, the two-target case with differing laws, and an auxiliary observation that carries no further target information.

Theorem 1 (Observation-equivalence limit)**.** *Let Z denote a latent target location and let O collect every observation available to a learner. Include any learner randomness independent of Z in O. (i) Let m be the number of candidate targets and suppose Z is uniform on $\{1, \dots, m\}$. Assume $\mathcal{L}(O \mid Z = z) = P$ for every z, where P is their common observation law. Then every possibly randomized estimator $\widehat{Z}(O)$ has $\Pr\{\widehat{Z} = Z\} \le 1/m$. (ii) For $m = 2$, index the equally likely candidates by zero and one and write their conditional laws as P_0 and P_1. The optimal success probability is $\sup_{\widehat{Z}} \Pr\{\widehat{Z} = Z\} = \big(1 + \mathrm{TV}(P_0, P_1)\big)/2$ (Yu, 1997). (iii) Let W be an auxiliary observation satisfying $Z \perp W \mid O$. Observing (O, W) gives the same Bayes success probability as observing O.*

Appendix A gives the formal WL and AUC definitions with the identification and learning-invariance proofs. The theorem concerns a target-generation model: success depends on how its observation laws distinguish the candidate locations. Under exact equivalence, improving on the uniform-prior chance rate requires additional target information. With unequal priors, choosing the most probable target can already exceed $1/m$.

4.2 From shared message passing to equal rankings

We characterize score equality by applying the standard WL characterization of message passing (Xu et al., 2019; Morris et al., 2019) to a shared node-scoring head.

Lemma 2 (Shared MPNN score equality)**.** *Consider a k-layer MPNN with shared message and update maps and permutation-invariant neighbour aggregation. Its input encoder is shared across nodes. A permutation-invariant readout produces a graph representation shared by all nodes, and a shared head scores each node from its final state and that representation. For every such network and every fixed graph, nodes with the same iteration-k WL colour have identical final states and identical scores. Hence its node scorer is $\le$1-WL representable.*

This is a representation-level equality for fixed inputs. A non-tie on equal-colour nodes indicates that some stated model or input condition is absent. It does not test the statistical observation-law premise of Theorem 1. For a fixed scorer, moving relevance within an equal-score class preserves every class-pair contribution to AUC when positive and negative counts stay fixed. This counting argument gives the next proposition.

Proposition 3 (Node-attribution barrier)**.** *Under Assumptions 1–2, every $\le$1-WL representable scorer has the same attribution AUC in the two worlds, giving*

$$\mathrm{AUC}(s, \ell^A) = \mathrm{AUC}(s, \ell^B). \tag{1}$$

This equality holds for a fixed scorer s and covers shared MPNN scores satisfying Lemma 2. Moreover, s is constant on each colour class. Its pair AUC on an automorphic witness pair therefore equals $\frac{1}{2}$ in both worlds. Under Assumption 3, these worlds have distinct node targets despite their identical observable data and ranking performance.

Overall AUC can reflect successful comparisons between different colour classes even when the witness pair is tied. Moving a relevant label within a class changes which node is correct without changing the scores compared in AUC, as Figure 1 shows on a test–statement graph. Overall AUC is therefore insensitive to these within-class target swaps for a fixed scorer. Pair AUC directly displays whether the specified witness pair is distinguished. The fixed-score statement extends to learning because identical training observations and world-independent randomness induce the same distribution of fitted scorers. Define the ambiguity class $\mathcal{A}(\ell)$ as all relevance assignments satisfying W1 and preserving the within-colour counts of ℓ in W2. Every member has the same AUC for a fixed $\leq$1-WL scorer. Appendix C gives its cardinality as a product of within-class choices and evaluates it on finite graphs.

4.3 Bounding optimal accuracy from observation counts

We now estimate a statistical accuracy bound from independent observations collected separately under each candidate target. This sampling design supplies conditional counts, whereas pooling observations across unknown targets does not. Appendix D describes the controlled channels used to evaluate this step.

Let $[m] = \{1, \ldots, m\}$ index targets and $[K] = \{1, \ldots, K\}$ a known, fixed alphabet of all possible observations, including unobserved types. For each z, the empirical law $\widehat{P}_z$ uses $n \geq 1$ independent samples from $P_z = \mathcal{L}(O \mid Z = z)$, for mn observations in total. For $K \geq 2$ and error probability $\alpha \in (0, 1)$, a union bound over the finite-alphabet TV inequality of Mardia et al. (2020, Lemma 1) gives radius

$$r(m, K, n, \alpha) = \sqrt{\frac{\log(2^K - 2) + \log(m/\alpha)}{2n}}.$$

For each observed value o, the optimal rule selects the target with largest $P_z(o)$. Summing these likelihoods and averaging over the uniform target prior gives the accuracy in the following bound.

Proposition 4 (Plug-in localization bound)**.** *Let Z be uniform on $[m]$ and let O take values in $[K]$, and write $A(P) = \frac{1}{m}\sum_o \max_z P_z(o)$ for the optimal localization accuracy. With $r = r(m, K, n, \alpha)$ as above, the event $\mathrm{TV}(\widehat{P}_z, P_z) \leq r$ for every z has probability at least $1 - \alpha$, and on that event*

$$A(P) \;\leq\; V_n := \min\big\{1,\; A(\widehat{P}) + r\big\}. \tag{2}$$

The empirical optimum gains one radius to cover sampling error, and clipping at one enforces the probability range. Averaging over target-specific decision regions incurs a single radius, as shown in Appendix A.9. For fixed m, K and α, it shrinks as $n^{-1/2}$. Changing or coarsening the observation alphabet changes the localizer's information: a bound for compressed observations applies to localizers using that compression. Known population equivalence instead gives the exact chance limit directly. Appendix A.6 derives the looser TV comparator U_n, and Appendix D gives both count-based computations.

5 Experiments

The experiments connect the theory to five questions: how often real inputs leave matched candidates, whether their scores tie as Lemma 2 predicts, which additional observations improve localization, how tightly finite counts bound optimal accuracy, and which natural targets require within-class distinctions. Real-corpus comparisons test representation equality. Controlled worlds and sampled observation laws test the statistical statements under their specified observation mechanisms.

5.1 Experimental setup

We study repaired statements in Defects4J Java faults (Just et al., 2014) and Codeflaws C faults (Tan et al., 2017). Coverage links tests to candidate statements. We first match repaired and non-repaired candidates by coverage, add local syntax descriptors, and retain pairs indistinguishable under feature-preserving colour refinement. Paths, line numbers and repair identity are withheld. Spectrum features use the same test outcomes that define the graph label. Consequently, high test-outcome AUROC measures the separation between graph prediction and node localization with these observations. Splits hold out faults, or an entire project, as specified below.

Table 1: Local descriptors reduce matched faults while leaving substantial within-input ambiguity. Counts are exact for each eligible population, before training. Parentheses give percentages of eligible faults.

Corpus	Eligible faults	Coverage only	Coverage + descriptors
Defects4J	330	270 (81.8%)	115 (34.8%)
Codeflaws	2,509	2,150 (85.7%)	830 (33.1%)

GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018) and a bipartite Transformer (Vaswani et al., 2017) learn from test outcomes. We compare their native node scores, gradient-times-input and feature occlusion (He et al., 2025; Shrikumar et al., 2017; Zeiler & Fergus, 2014) with coverage-based Ochiai, DStar and Tarantula (Xie et al., 2013; Abreu et al., 2007; Wong et al., 2014; Jones & Harrold, 2005). The oracle target anchor adds a coordinate indicating the repaired node. It supplies target information for the intervention comparison.

Test-outcome AUROC ranks failed tests, pooled node AUROC ranks repaired statements across the evaluated faults, and pair AUC restricts ranking to matched repaired/non-repaired pairs. A tie contributes 0.5. Maximum score gap measures numerical departure from equality, in the scorer's raw units. Neural entries report mean $\pm$ sample SD over three seeds unless stated otherwise. Corpus counts and spectrum scores are deterministic, and the Defects4J GNNExplainer comparison is a single evaluation. Table 15 defines the remaining measures, and Appendix H gives training and selection settings.

5.2 Matched candidates in two real corpora

Does adding local syntax resolve the matches left by coverage? Table 1 shows that it removes many matches while leaving roughly one third of eligible faults with at least one matched pair. Eligibility requires an observed repaired statement and competing non-repaired candidates. Codeflaws additionally requires a single repaired old line and both passing and failing tests. The full filtering and neural split counts appear in Table 23.

Additional tests also resolve some matches. On the same 2,349 Codeflaws faults, expanding the shipped suite reduces the matched fraction from 34.5% to 31.9%: 60 faults lose their match and none gains one (Table 26). Thus the remaining ambiguity depends on the supplied observations. Matched faults span five Defects4J projects and 550 Codeflaws problems, including 95 Defects4J faults with matches beyond the nearby-statement relation defined in Appendix I.1.

These ties have a measurable consequence for a class-constant ranking. An oracle choosing the best tied class and then ordering its members uniformly has mean top-1 success 0.794 on eligible Codeflaws faults, falling to 0.377 on the matched subset (Table 24). This is a ceiling for that ranking protocol. Multiple repaired statements can provide alternative successful targets, explaining the higher corresponding Defects4J values.

5.2.1 Java faults: graph prediction and pair localization

The Defects4J neural population has 255 faults: 172 for training, 33 for validation, 31 held-out fault roots and 19 faults in the held-out Time project. Table 2 shows high test-outcome AUROC alongside pair AUC 0.5 in both evaluation splits. The evaluated shared heads assign equal scores to matched representations, directly testing Lemma 2.

Independent node-mask optimization can break those score ties. A single GNNExplainer evaluation (Ying et al., 2019) separates all 25 root-held-out pairs, but correctly ranks only nine. The fault-bootstrap pair-AUC interval is $[0.19, 0.50]$, using the 15 faults that supply these pairs (Table 18). The result distinguishes score separation from successful localization. It does not identify the cause of every score difference. On the same held-out candidate sets, the three spectrum methods also tie every matched pair despite differing pooled node AUROCs (Table 21).

5.2.2 C faults: equality across attribution rules

Codeflaws tests the same representation mechanism in a second workload. All nine combinations of the three architectures and three attribution rules have pair AUC 0.5 on 1,986 matched pairs, including

Table 2: The evaluated Defects4J models predict test outcomes while tying matched candidates. Means ± sample SD over three seeds. Root-held-out evaluation has 31 faults, 4,279 candidates and 25 pairs. The held-out Time project has 19 faults, 4,342 candidates and 231 pairs. The visible-score equality check uses CPU float64 evaluation; pair AUC is 0.500 in both splits for every run.

model	root test AUROC ↑	project test AUROC ↑	root node AUROC ↑	root pair AUC ↑
GraphSAGE	0.994 ± 0.001	0.984 ± 0.003	0.627 ± 0.012	0.500 ± 0.000
GAT	0.991 ± 0.003	0.981 ± 0.001	0.741 ± 0.024	0.500 ± 0.000
Transformer	0.998 ± 0.001	0.992 ± 0.002	0.742 ± 0.016	0.500 ± 0.000

Table 3: Codeflaws observation changes, using GraphSAGE native scores. All ranking entries are mean ± sample SD over three seeds. Test and pooled node AUROC use held-out faults. The 282 held-out pairs determine localization performance; the 1,986 all-pair results, including training pairs, are separated in Table 25.

Observation	Test AUROC	Node AUROC	Held-out pair AUC
Original	0.884 ± 0.001	0.619 ± 0.012	0.500 ± 0.000
Random coordinates	0.881 ± 0.001	0.641 ± 0.007	0.467 ± 0.024
Target anchor	0.884 ± 0.001	0.698 ± 0.040	0.960 ± 0.063

training pairs (Table 30). The largest absolute gap is 1.85×10^{-13}, below the 10^{-9} tie tolerance. This all-pair evaluation checks structural score equality. The held-out localization comparison below uses the separate subset of 282 pairs.

5.3 Changing the observations

Does distinguishing node identities provide the information needed to identify the repaired node? We append either eight target-independent Gaussian coordinates or the oracle target anchor. Each random coordinate vector is fixed within a training run, with corpus-specific seeding described in Appendix J. Table 3 shows that random coordinates separate scores without a consistent held-out localization gain. The target anchor raises held-out pair AUC to 0.960, with visible seed variation. Information availability and a trained model's use of it are therefore separate questions.

The Defects4J random-coordinate comparison varies across architectures and splits (Table 20). Its project-held-out mean over nine runs is 0.467, compared with 0.416 for GraphSAGE alone. These realized rankings measure the effect of the sampled coordinates. Theorem 1(iii) separately establishes the population result when the added feature is conditionally independent of the target. Raw score gaps and the complete-node-truth oracle are reported in Tables 27 and 29.

5.4 Information, estimation and target definition

5.4.1 Separating anchor information from supervision

Executed Boolean circuits and Python ASTs provide controlled faulty-node targets. Their base comparison adds an anchor while retaining graph-only training, giving within-orbit AUC 1.0 in the anchored groups (Table 8). The broader architecture extension supplies both anchor and node supervision in its target-escape setting (Table 10). To isolate these two ingredients, the EPFL factorial crosses them on 1,200 executed component faults from seven unmodified designs (Amarú et al., 2015), using GIN and GraphSAGE with random and module-held-out splits.

Table 4 shows that node supervision alone leaves within-orbit ranking near chance, whereas the anchor supports high accuracy. Each cell averages 12 original runs. An expanded anchor-only batch uses six seeds per architecture–split group and has overall mean 0.956 (Table 9). One GIN module-held-out run has pair AUC 0.392 despite graph accuracy 0.969. Thus an available target distinction raises attainable accuracy while its learned use can remain unstable. The approximate-symmetry and query comparisons in Tables 11 and 12 further separate arbitrary representation changes from target-bearing information.

Table 4: EPFL factorial separates target information from node supervision. Each cell contains three seeds for each of two architectures and two splits (12 runs). The range is across the four group means, not individual seeds.

Target anchor	Node supervision	Mean orbit AUC	Group-mean range
Absent	Absent	0.502	[0.475, 0.522]
Absent	Present	0.511	[0.495, 0.538]
Present	Absent	0.980	[0.961, 0.999]
Present	Present	1.000	[1.000, 1.000]

Table 5: Main bound V_n and TV comparator U_n on the same 10,000 sampled channels ($K = 4$, $\alpha = 0.05$). Excess is measured above optimal accuracy $A(P)$, distinct from $U_n - B(P)$ in Table 7. Each sample size aggregates the three candidate counts. Both bounds cover $A(P)$ in every trial.

n per target	Trials	Median $V_n - A$	Median $U_n - A$	P90 $V_n - A$	P90 $U_n - A$
100	3334	0.204	0.401	0.226	0.465
500	3333	0.090	0.177	0.100	0.236
2000	3333	0.044	0.088	0.049	0.150

5.4.2 Finite counts bound optimal accuracy

We next test Proposition 4 on known discrete observation laws, where optimal accuracy $A(P)$ can be computed. The 10,000 trials span $m \in \{2, 4, 8\}$ targets and $n \in \{100, 500, 2000\}$ samples per target, with known alphabet size $K = 4$ and error probability 0.05. Table 5 compares V_n and the TV relaxation U_n against the same $A(P)$. Both cover every sampled channel, and increasing n reduces median and upper-tail excess. The tighter V_n avoids the additional population relaxation in U_n. This small-alphabet experiment evaluates statistical estimation. Real-corpus score equality has the separate role established above, since the complete coverage alphabet can be much larger (Appendix D).

5.4.3 Which natural targets require within-class distinctions?

Common explanation benchmarks test the scope of the within-class target assumption. We count colour classes containing both relevant and irrelevant nodes under their natural masks. Table 13 records 13 affected trees among 9,129 eligible sentiment trees at depth two (Socher et al., 2013), and no affected graph at depth four in the six evaluated molecular or motif datasets (Agarwal et al., 2023; Debnath et al., 1991; Kazius et al., 2005; Morris et al., 2020; Luo et al., 2020).

Natural masks can therefore use distinctions between classes. In the original random-split comparisons, the best architecture–attribution mean natural-mask AUC is 0.961 on Mutagenicity and 0.832 on Fluoride-Carbonyl (Table 14). Separately constructed targets that exchange members under fixed observations yield paired AUC 0.5. These are different target definitions: successful natural-mask recovery can coexist with the inability to recover an exchanged within-class target. Structure-held-out evaluations and additional seeds retain this distinction.

6 Conclusion

In this paper, we establish that equally likely node targets with identical observation laws have chance-level optimal localization accuracy regardless of which model is trained on those observations. We characterize relevance assignments that preserve rankings for shared message-passing scorers and bound localization accuracy from conditional observation counts. Matched repaired and non-repaired statements remain common in real software after coverage is enriched with local descriptors, and shared-head scorers tie on them. The circuit comparisons separate target information from node supervision and show how one target-bearing coordinate enables accurate pair localization. Molecular controls recover natural masks while exchanged within-class targets yield chance pair AUC. These findings support evaluating graph prediction and node localization against the specific target and the observations provided for attribution. Which observations are cheap enough to collect and rich enough to identify the responsible node remains an open question.

AI USE STATEMENT

In this work, we used generative AI tools for tasks with required disclosure. Generative AI tools helped develop the theoretical models and conceptual frameworks, helped formulate the mathematical claims, provided critical ingredients for proving those claims, and assisted in the writing of the proofs. The mathematical claims, the proofs, and every experimental result were checked and verified by the authors. We additionally used generative AI tools for tasks with recommended disclosure, namely to aid and polish the writing and to edit the paper for readability. We have reviewed all AI-assisted work, including the mathematical derivations and the text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

[Statement to be written and confirmed by the authors before submission.]

REPRODUCIBILITY STATEMENT

The graph constructions and discrete-channel evaluations use NumPy and NetworkX with SciPy and scikit-learn (Harris et al., 2020; Hagberg et al., 2008; Pedregosa et al., 2011). The defect-corpus descriptions give eligibility conditions and feature definitions together with evaluation splits. Codeflaws coverage is collected using `gcc --coverage` and `gcov`. Appendix H reports real-corpus optimizer settings and validation-based model selection with per-run training lengths. The GNNExplainer configuration and matched-candidate spectrum comparisons are in Appendix I. Proofs and the complete experimental comparisons follow below.

REFERENCES

Rui Abreu, Peter Zoeteweij, and Arjan J.C. van Gemund. On the Accuracy of Spectrum-based Fault Localization. In *Testing for DR: Academic and Industrial Conference Practice and Research Techniques - MUTATION (TAICPART-MUTATION 2007)*, pp. 89–98, 2007. doi: 10.1109/taic.part.2007.13.

Chirag Agarwal, Marinka Zitnik, and Himabindu Lakkaraju. Probing GNN explainers: A rigorous theoretical and empirical analysis of GNN explanation methods. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 8969–8996, 2022.

Chirag Agarwal, Owen Queen, Himabindu Lakkaraju, and Marinka Zitnik. Evaluating explainability for graph neural networks. *Scientific Data*, 10(1):144, 2023.

Luca Amarú, Pierre-Emmanuel Gaillardon, and Giovanni De Micheli. The EPFL combinational benchmark suite. In *Proceedings of the 24th International Workshop on Logic Synthesis (IWLS)*, 2015.

Steve Azzolin, Sagar Malhotra, Andrea Passerini, and Stefano Teso. Beyond topological self-explainable GNNs: A formal explainability perspective, 2025. arXiv preprint arXiv:2502.02719.

Steve Azzolin, Stefano Teso, Bruno Lepri, Andrea Passerini, and Sagar Malhotra. GNN explanations that do not explain and how to find them. In *International Conference on Learning Representations (ICLR)*, 2026. URL `https://openreview.net/forum?id=HBcgLe6NZD`.

Benoit Baudry, Franck Fleurey, and Yves Le Traon. Improving test suites for efficient fault localization. In *Proceedings of the 28th International Conference on Software Engineering (ICSE)*, pp. 82–91, 2006. doi: 10.1145/1134285.1134299.

Visar Berisha, Alan Wisler, Alfred O. Hero, and Andreas Spanias. Empirically estimable classification bounds based on a nonparametric divergence measure. *IEEE Transactions on Signal Processing*, 64(3):580–591, 2016. doi: 10.1109/TSP.2015.2477805.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations (ICLR)*, 2022.

Ruichu Cai, Yuxuan Zhu, Xuexin Chen, Yuan Fang, Min Wu, Jie Qiao, and Zhifeng Hao. On the probability of necessity and sufficiency of explaining graph neural networks: A lower bound optimization approach, 2022. arXiv preprint arXiv:2212.07056.

José Campos, André Riboira, Alexandre Perez, and Rui Abreu. GZoltar: an eclipse plug-in for testing and debugging. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pp. 378–381, 2012. doi: 10.1145/2351676.2351752.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal Neighbourhood Aggregation for Graph Nets, 2020. URL https://arxiv.org/abs/2004.05718.

Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, 1991. doi: 10.1021/jm00106a046.

Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1–2):31–71, 1997.

Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric, 2019. URL https://arxiv.org/abs/1903.02428.

Michele Fontanesi, Alessio Micheli, Marco Podda, and Domenico Tortorella. A method for the systematic generation of graph XAI benchmarks via Weisfeiler–Leman coloring. *Data Mining and Knowledge Discovery*, 40(4):42, 2026. doi: 10.1007/s10618-026-01212-z.

Alberto Gonzalez-Sanchez, Rui Abreu, Hans-Gerhard Gross, and Arjan J. C. van Gemund. Prioritizing tests for fault localization through ambiguity group reduction. In *Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 83–92, 2011. doi: 10.1109/ASE.2011.6100153.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy)*, pp. 11–15, 2008.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 2017.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, et al. Array programming with NumPy. *Nature*, 585:357–362, 2020.

Jesse He, Akbar Rafiey, Gal Mishne, and Yusu Wang. Explaining GNN explanations with edge gradients. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, 2025.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi: 10.1080/01621459.1963.10500830.

Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 2127–2136, 2018.

James A. Jones and Mary Jean Harrold. Empirical evaluation of the tarantula automatic fault-localization technique. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, pp. 273–282, 2005. doi: 10.1145/1101908.1101949.

René Just, Darioush Jalali, and Michael D. Ernst. Defects4J: A database of existing faults to enable controlled testing studies for Java programs. In *Proceedings of the 2014 International Symposium on Software Testing and Analysis*, pp. 437–440. ACM, 2014. doi: 10.1145/2610384.2628055.

Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and Validation of Toxicophores for Mutagenicity Prediction. *Journal of Medicinal Chemistry*, 48(1):312–320, 2005. doi: 10.1021/jm040835a.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yiling Lou, Qihao Zhu, Jinhao Dong, Xia Li, Zeyu Sun, Dan Hao, Lu Zhang, and Lingming Zhang. Boosting coverage-based fault localization via graph-based representation learning. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 664–676, 2021. doi: 10.1145/3468264.3468580.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Jay Mardia, Jiantao Jiao, Ervin Tánczos, Robert D Nowak, and Tsachy Weissman. Concentration inequalities for the empirical distribution of discrete distributions: beyond the method of types. *Information and Inference: A Journal of the IMA*, 9(4):813–850, 2020. doi: 10.1093/imaiai/iaz025.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Edward J. McCluskey and Frederick W. Clegg. Fault equivalence in combinational logic networks. *IEEE Transactions on Computers*, C-20(11):1286–1293, 1971. doi: 10.1109/T-C.1971.223129.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs, 2020. URL https://arxiv.org/abs/2007.08663.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer, 2022. URL https://arxiv.org/abs/2205.12454.

Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016.

Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T. Schutt, Klaus-Robert Muller, and Gregoire Montavon. Higher-Order Explanations of Graph Neural Networks via Relevant Walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7581–7596, 2022. doi: 10.1109/tpami.2021.3115452.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3145–3153. PMLR, 2017. URL https://proceedings.mlr.press/v70/shrikumar17a.html.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, 2013.

Friedrich Steimann, Marcus Frenkel, and Rui Abreu. Threats to the validity and value of empirical assessments of the accuracy of coverage-based fault locators. In *Proceedings of the 2013 International Symposium on Software Testing and Analysis (ISSTA)*, pp. 314–324, 2013. doi: 10.1145/2483760.2483767.

Wenxin Tai, Ting Zhong, Goce Trajcevski, and Fan Zhou. Redundancy Undermines the Trustworthiness of Self-Interpretable GNNs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 58169–58188. PMLR, 2025. URL https://proceedings.mlr.press/v267/tai25a.html.

Wenxin Tai, Yaqian Liu, Ting Zhong, and Fan Zhou. Why Self-Inconsistency Arises in GNN Explanations and How to Exploit It, 2026. URL https://arxiv.org/abs/2605.07527.

Shin Hwei Tan, Jooyong Yi, Yulis, Sergey Mechtaev, and Abhik Roychoudhury. Codeflaws: A programming competition benchmark for evaluating automated program repair tools. In *Proceedings of the 39th International Conference on Software Engineering Companion (ICSE-C)*, 2017.

Ramesh C. Tekumalla and Scott Davidson. On identifying indistinguishable path delay faults and improving diagnosis. In *Proceedings of the International Test Conference (ITC)*, pp. 993–1002, 2002. doi: 10.1109/TEST.2002.1041855.

Ryan Theisen, Huan Wang, Lav R. Varshney, Caiming Xiong, and Richard Socher. Evaluating state-of-the-art classification models against Bayes optimality, 2021. arXiv preprint arXiv:2106.03357.

Ming Tu, Jing Huang, Xiaodong He, and Bowen Zhou. Multiple instance learning with graph neural networks, 2019. ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Representations; arXiv:1906.04881.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

John A. Waicukauski and Eric Lindbloom. Failure diagnosis of structured VLSI. *IEEE Design & Test of Computers*, 6(4):49–60, 1989. doi: 10.1109/54.32421.

Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

W. Eric Wong, Vidroha Debroy, Ruizhi Gao, and Yihao Li. The DStar Method for Effective Software Fault Localization. *IEEE Transactions on Reliability*, 63(1):290–308, 2014. doi: 10.1109/tr.2013.2285319.

Xiaoyuan Xie, Tsong Yueh Chen, Fei-Ching Kuo, and Baowen Xu. A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization. *ACM Transactions on Software Engineering and Methodology*, 22(4), 2013. doi: 10.1145/2522920.2522924.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Zimo Yan, Zheng Xie, Chang Liu, and Yuan Wang. Resolving node identifiability in graph neural processes via laplacian spectral encodings, 2025. arXiv preprint arXiv:2511.19037.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Bin Yu. Assouad, Fano, and Le Cam. In *Festschrift for Lucien Le Cam*, pp. 423–435. Springer New York, 1997. doi: 10.1007/978-1-4612-1880-7_29.

Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 12241–12252, 2021.

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5): 5782–5799, 2023.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *Computer Vision – ECCV 2014*, pp. 818–833. Springer International Publishing, 2014. doi: 10.1007/978-3-319-10590-1_53.

A Proofs and auxiliary definitions

This appendix supports Sections 3 and 4 with the ranking definition and proofs of the identification results.

A.1 WL definitions and candidate worlds

Definitions 5 and 6 formalize the representation descriptions used in Section 3.

Definition 5 (Iteration-k 1-WL colours)**.** Let k be a fixed nonnegative integer specifying the refinement depth. Initialize $c_G^{(0)}(v) = X(v)$, so iteration zero distinguishes only the input types. For iterations $t = 1, \dots, k$, update the colour by combining its previous value with the neighbouring colours as

$$c_G^{(t)}(v) = \mathrm{hash}\Big(c_G^{(t-1)}(v),\ \{\{c_G^{(t-1)}(u) : u \in N(v)\}\}\Big), \tag{3}$$

where $\{\{\cdot\}\}$ denotes a multiset that retains repeated colours. The map hash assigns different identifiers to distinct colour–neighbour-multiset pairs. Thus nodes receive the same new colour exactly when both descriptions agree. We abbreviate $c_G^{(k)}(v)$ as $c(v)$. A colour class C contains all nodes with a given colour in a fixed graph.

Definition 6 ($\leq$1-WL representable scorer)**.** A node scorer s is *$\leq$1-WL representable at depth k* if equal iteration-k colours imply equal scores within every fixed graph G. Formally, $c_G^{(k)}(u) = c_G^{(k)}(v) \Rightarrow s(G,u) = s(G,v)$. Equivalently, $s(G,v) = \bar{s}_G(c_G^{(k)}(v))$ for a graph-indexed colour-to-score map $\bar{s}_G$.

Tie-aware attribution AUC. Given scores $s(v)$ and a binary relevance labelling $\ell(v)$ over a specified finite node collection (a single graph or the disjoint union of a fixed graph collection) with $n_y = |\{v : \ell(v) = y\}|$,

$$\mathrm{AUC}(s,\ell) = \frac{1}{n_1 n_0} \sum_{v:\ell(v)=1} \sum_{u:\ell(u)=0} \Big[\mathbf{1}\{s(v)>s(u)\} + \tfrac{1}{2}\mathbf{1}\{s(v)=s(u)\}\Big]. \tag{4}$$

A.2 Proof of Theorem 1

This section proves the observation-equivalence limit in Section 4.1.

Proof of Theorem 1. For (i), let $q_z(o)$ be the conditional probability that the estimator outputs z. Under the common law P, $\Pr\{\widehat{Z} = Z\} = \frac{1}{m}\int \sum_{z=1}^{m} q_z(o)\, dP(o) \leq \frac{1}{m}$. For (ii), with two equal-prior hypotheses the maximizing decision region is the positive part of the signed measure $P_1 - P_0$, and the resulting success probability is the standard binary testing identity. For (iii), $Z \perp W \mid O$ gives $\Pr(Z = z \mid O, W) = \Pr(Z = z \mid O)$ almost surely. The maximizing posterior probability is therefore unchanged, as is Bayes success. □

A.3 Proof of Proposition 3

This section proves the ranking equality in Section 4.2.

Proof of Proposition 3. Since s is $\leq$1-WL representable, $s(G,v) = \bar{s}_G(c(v))$ depends on v within a fixed graph only through its colour. Treat each graph–colour pair $a = (G,c)$ as an atom, write

$C_a = \{v \in V(G) : c_G(v) = c\}$, and define $N_\ell(a, y) = \#\{v \in C_a : \ell(v) = y\}$ for $y \in \{0, 1\}$. Grouping the double sum in (4) by the atoms of the positive and negative nodes gives

$$\mathrm{AUC}(s, \ell) = \frac{1}{n_1 n_0} \sum_{a,a'} N_\ell(a, 1) N_\ell(a', 0) \Big[\mathbf{1}\{\bar{s}(a) > \bar{s}(a')\} + \tfrac{1}{2} \mathbf{1}\{\bar{s}(a) = \bar{s}(a')\} \Big], \tag{5}$$

with $n_y = \sum_a N_\ell(a, y)$ and $\bar{s}(a) := \bar{s}_G(c)$. By Assumption 2 the relevant count per colour class is world-invariant on each graph, so $N_{\ell^A}(a, 1) = N_{\ell^B}(a, 1)$ for every atom. The total per atom is fixed by the world-independent graph structure, so $N_{\ell^A}(a, 0) = N_{\ell^B}(a, 0)$ as well, and $N_{\ell^A} \equiv N_{\ell^B}$ elementwise. The map $\bar{s}$ is one and the same scorer in both worlds, so the right-hand side of (5) takes the same value under ℓ^A and ℓ^B, which is (1). For the twin claim, let C be a colour class in G containing an automorphic pair w_1, w_2. Both share the colour c, so $s(G, w_1) = s(G, w_2) = \bar{s}_G(c)$. The scores are tied. Whether the truly relevant twin is w_1 or w_2, the tie contributes $\frac{1}{2}$ to pair AUC, which is therefore $\frac{1}{2}$ in both worlds. □

A.4 Learning under identical observations

This section supports Section 4.2 by relating the fixed-score result to the distribution of trained scorers.

The proposition concerns a fixed scorer. To connect it to training, we also establish that the observations yield the same distribution of learned scorers across worlds.

Corollary 7 (Learning invariance)**.** *Under Assumptions 1–2, let a learning rule produce s from $(G, Y(G))_{G \in \mathcal{G}}$. Its internal randomness has a fixed law independent of the world. Suppose its node scores are evaluated through $\leq$1-WL representations. Then the output law is world-independent and its attribution AUC obeys Eq.* (1)*.*

The proof applies the learning rule to the same data and randomness in both worlds. Each realized scorer then satisfies Proposition 3. Training can exploit graph-level signal while preserving the ambiguity in the node target. The controlled experiments in Appendix E evaluate this separation.

Proof of Corollary 7. Write a run of the rule as $s = \Lambda(D, U)$ with measurable Λ. Here $D = (G, Y(G))_{G \in \mathcal{G}}$ is the observable data. The rule's internal randomness U has a fixed law μ independent of the world. By Assumption 1 the graphs are world-independent and Y is a function of the graph, so D is one and the same object in world A and in world B. The latent labelling ℓ enters neither argument of Λ. Hence for every measurable set S of scorers, $\Pr_A\{s \in S\} = \int \mathbf{1}\{\Lambda(D, u) \in S\}\, d\mu(u) = \Pr_B\{s \in S\}$. The two integrals have identical integrands and measures. The output law is therefore world-independent. Each realization s is $\leq$1-WL representable by hypothesis, so Proposition 3 applies to it pathwise and gives $\mathrm{AUC}(s, \ell^A) = \mathrm{AUC}(s, \ell^B)$ for every draw of U, hence also for the induced joint law of the two AUCs. □

A.5 Shared message passing and readout structure

This section supports the shared-score statement in Section 4.2 through the full colour-atom reduction.

The next lemma makes this connection explicit and identifies the graph information retained by a decomposable readout.

Lemma 8 (Colour-atom reduction)**.** *Consider a k-layer message-passing neural network (MPNN) with node state $h_v^{(t)}$ at layer t. A shared input encoder E_{emb} initializes $h_v^{(0)} = E_{\mathrm{emb}}(X(v))$. At each layer, a message map M_t and aggregation map AGG_t compute $m_v^{(t)} = \mathrm{AGG}_t\{\{M_t(h_v^{(t-1)}, h_u^{(t-1)}) : u \in N(v)\}\}$. Here $m_v^{(t)}$ is the aggregated message received by node v. The update map then sets $h_v^{(t)} = \mathrm{UPD}_t(h_v^{(t-1)}, m_v^{(t)})$. All these maps are shared across nodes and AGG_t is permutation-invariant. A permutation-invariant readout R produces the graph representation $h_G = R(\{\{h_v^{(k)}\}\})$. A shared head S assigns $s(G, v) = S(h_v^{(k)}, h_G)$. Write $h_v = h_v^{(k)}$ for the final node state.*

(i) There is a map f_k with $h_v = f_k(c_G^{(k)}(v))$. Hence s is $\leq$1-WL representable within each graph. (ii) Suppose the readout decomposes as $R(G) = \phi(\sum_v \psi(h_v))$, where $R(G)$ abbreviates the readout applied to the node states of G. Here ψ transforms individual node states and ϕ maps their sum to

the graph representation. Writing $\tilde{\psi} = \psi \circ f_k$, we can express this readout using colour counts as

$$R(G) = \phi\Big(\sum_c n_c(G)\,\tilde{\psi}(c)\Big), \qquad n_c(G) = \#\{v : c_G^{(k)}(v) = c\}, \tag{6}$$

where $n_c(G)$ counts the nodes with colour c. Each colour contributes its transformed representation multiplied by its frequency in the graph. (iii) For graphs with at most N nodes and initial types in the finite alphabet Σ_0, the reachable iteration-k colour set is finite. Here N is an upper bound on graph size.

This section proves the lemma by induction over layers and then groups the readout terms by colour. The lemma states that equal colours yield equal node states and equal scores even when the scoring head also uses graph context. Shared maps and permutation-invariant aggregation preserve this equality. Decomposability is required only for the histogram identity in Eq. (6). That identity is the DeepSets sum representation (Zaheer et al., 2017) applied to WL colours, with their counts supplying the multiplicities. Mean pooling replaces each count $n_c(G)$ by its relative frequency $n_c(G)/|V|$. Appendix C gives the corresponding finite-graph evaluation.

Proof of Lemma 8. (i) Induct on t. At $t = 0$, equal colours mean equal discrete features, so shared E_{emb} gives $h_v^{(0)} = f_0(c_G^{(0)}(v))$. Assume $h_v^{(t-1)} = f_{t-1}(c_G^{(t-1)}(v))$. Then the aggregated multiset is a function of $c_G^{(t-1)}(v)$ and $\{\{c_G^{(t-1)}(u) : u \in N(v)\}\}$, so $h_v^{(t)}$ is a function of that pair, which by (3) determines $c_G^{(t)}(v)$. The map hash is injective on distinct such pairs and is therefore a bijection onto its image. Thus $c_G^{(t)}(v)$ determines that pair and hence $h_v^{(t)}$. Composing the two maps gives a well-defined f_t. At $t = k$ this is the claim. Within a fixed graph h_G is shared by all nodes, so equal final colours imply equal $S(h_v^{(k)}, h_G)$. *(ii)* By (i), $\psi(h_v) = \tilde{\psi}(c_G^{(k)}(v))$. Grouping the sum by colour gives (6), and mean-pool replaces $n_c(G)$ by $n_c(G)/|V|$. *(iii)* The iteration-k colour of v is determined by the depth-k rooted WL unfolding labelled by initial types (Morris et al., 2019; Xu et al., 2019). On graphs with at most N nodes the degree is at most $N - 1$ and the number of distinct depth-k labelled rooted unfoldings is bounded by a finite function of N, k and $|\Sigma_0|$. □

A.6 A TOTAL-VARIATION RELAXATION FOR COMPARISON

This section derives a comparator for the main bound V_n in Section 4.3. It first upper-bounds the optimal accuracy $A(P)$ by a TV relaxation $B(P)$, then estimates that relaxation with an additional sampling allowance.

For multiple targets we compare their observation laws with a uniform mixture.

Proposition 9 (Multi-state bound)**.** *Let Z be uniform on $[m]$ and let O take values in $[K]$. Write $P_z = \mathcal{L}(O \mid Z = z)$ and $\bar{P} = \frac{1}{m}\sum_z P_z$ for the conditional laws and their mixture. The optimal localization accuracy and its TV upper bound satisfy*

$$\sup_{\widehat{Z}} \Pr\{\widehat{Z} = Z\} = \frac{1}{m}\sum_o \max_z P_z(o) \;\le\; \frac{1 + \sum_{z=1}^m \mathrm{TV}(P_z, \bar{P})}{m} =: B(P), \tag{7}$$

where $P_z(o)$ is the probability of observation o under target z.

The chance term $1/m$ and the TV terms separate prior uncertainty from target-dependent observation information. If every candidate produces the same observations, all TV terms vanish. If the observations occur at different rates under different targets, those differences can help select the responsible member. The mixture supplies a common reference for measuring each candidate's contribution. The left-hand side of Eq. (7) is the quantity $A(P)$ of Proposition 4, so $A(P) \le B(P)$ always holds, with equality at $m = 2$. The numerical comparison of $A(P)$ and $B(P)$ is reported with the sampled-channel results in Appendix D.

Estimating both the conditional laws and their mixture gives the following TV comparator.

Definition 10 (Total-variation certificate)**.** Let $\widehat{\bar{P}} = \frac{1}{m}\sum_z \widehat{P}_z$ be the empirical mixture. For the samples and confidence level of Section 4.3, define

$$U_n = \min\Big\{1,\; \frac{1 + \sum_{z=1}^m \mathrm{TV}(\widehat{P}_z, \widehat{\bar{P}})}{m} + 2\,r(m, K, n, \alpha)\Big\}, \tag{8}$$

where $2r$ covers estimation errors in the conditional laws and their mixture.

Proposition 11 (Validity)**.** *Under the sampling conditions above, the following holds with probability at least* $1-\alpha$ *over the samples used to construct* U_n*. Every estimator of* Z *has population success probability at most* U_n*.*

Because $A(\widehat{P}) \le B(\widehat{P})$ for every tuple of empirical laws, U_n is never below the V_n of Eq. (2). Thus $V_n \le U_n$ holds for every count table. Table 5 quantifies the resulting difference in excess over optimal accuracy on the same sampled channels.

A.7 Proof of Proposition 9

This section proves the population bound stated above.

Proof of Proposition 9. The Bayes rule outputs $\arg\max_z P_z(o)$, giving the stated success probability. For every o, $\max_z P_z(o) \le \bar{P}(o) + \sum_z (P_z(o) - \bar{P}(o))_+$. Summing over o and using $\sum_o (P_z(o) - \bar{P}(o))_+ = \mathrm{TV}(P_z, \bar{P})$ gives $\sum_o \max_z P_z(o) \le 1 + \sum_z \mathrm{TV}(P_z, \bar{P})$. At $m = 2$ the inequality is an identity because $\max(a,b) = \frac{a+b}{2} + \frac{|a-b|}{2}$ and $\mathrm{TV}(P_1, \bar{P}) = \mathrm{TV}(P_2, \bar{P}) = \frac{1}{2}\mathrm{TV}(P_1, P_2)$. If all P_z coincide, every total variation vanishes and both sides equal $1/m$. □

A.8 Proof of Proposition 11

This section proves the confidence statement for Definition 10.

Proof of Proposition 11. Write $\mathrm{TV} = \frac{1}{2}\|\cdot\|_1$. For fixed z, $\|\widehat{P}_z - P_z\|_1 = 2\max_{S\subseteq[K]}(\widehat{P}_z(S) - P_z(S))$ and there are $2^K - 2$ proper nonempty events S. Each $\widehat{P}_z(S)$ is an average of n i.i.d. Bernoulli variables, so Hoeffding's inequality (Hoeffding, 1963) and a union bound over those events give $\Pr\{\|\widehat{P}_z - P_z\|_1 \ge \varepsilon\} \le (2^K - 2)e^{-n\varepsilon^2/2}$. Taking $\varepsilon = 2r$ makes the right-hand side α/m. A union bound over the m latent states gives the event $\mathcal{E} = \{\mathrm{TV}(\widehat{P}_z, P_z) \le r \ \forall z\}$ with probability at least $1-\alpha$. On $\mathcal{E}$, convexity gives $\mathrm{TV}(\bar{P}, \widehat{\bar{P}}) \le r$, hence $\mathrm{TV}(P_z, \bar{P}) \le \mathrm{TV}(\widehat{P}_z, \widehat{\bar{P}}) + 2r$ for every z by the triangle inequality. Substituting into (7) bounds $B(P)$ by the right-hand side of (8), and Proposition 9 bounds the Bayes success probability by $B(P)$. □

A.9 Proof of Proposition 4

This section proves the plug-in bound in Section 4.3.

Proof of Proposition 4. The proof of Proposition 11 establishes that the event $\mathcal{E} = \{\mathrm{TV}(\widehat{P}_z, P_z) \le r \ \forall z\}$ has probability at least $1-\alpha$. A possibly randomized estimator of Z is determined by a partition $(D_z)_{z\in[m]}$ of $[K]$ together with the randomization already absorbed into O, and its success probability is $\frac{1}{m}\sum_z P_z(D_z)$. On $\mathcal{E}$ the definition of total variation gives $P_z(D_z) \le \widehat{P}_z(D_z) + r$ for every z, so

$$\frac{1}{m}\sum_z P_z(D_z) \;\le\; \frac{1}{m}\sum_z \widehat{P}_z(D_z) + r \;=\; \frac{1}{m}\sum_z \sum_{o\in D_z} \widehat{P}_z(o) + r \;\le\; \frac{1}{m}\sum_o \max_z \widehat{P}_z(o) + r,$$

where the last step uses that the sets D_z are disjoint, so each observation o contributes at most $\max_z \widehat{P}_z(o)$. The right-hand side is $A(\widehat{P}) + r$. Taking the supremum of the left-hand side over partitions gives $A(P)$, and any accuracy is at most one, which is Eq. (2). □

B EXTENDED RELATED WORK AND POSITIONING

This appendix supports Section 2 by comparing the objects studied by neighbouring research lines.

Table 6 states the object and mechanism of each neighbouring line of work and the corresponding difference.

Table 6: Positioning against the nearest neighbours. Our object is the data-law identifiability of a latent node-relevance target under decoupled graph-level supervision. This concerns the observation law before model expressivity or explainer reliability. It also differs from causal-scalar identifiability and self-explaining semantics.

neighbour	its object / mechanism	how this paper differs
WL / GNN expressivity (Weisfeiler & Leman, 1968; Morris et al., 2019; Xu et al., 2019; Maron et al., 2019; Bevilacqua et al., 2022)	can an architecture *distinguish* two non-isomorphic graphs (a ceiling on representational power)	we fix the data-generating law and the latent annotation and ask which annotations the observable law admits. Observational equivalence is architecture-independent. The score-tie and AUC statements additionally assume a WL-bounded score.
GNN explainer reliability (Agarwal et al., 2022)	upper bounds on the faithfulness / stability an explainer can violate	we ask whether the observable law identifies the node-relevance target independently of the explainer.
PNS non-identifiability (Cai et al., 2022)	a causal scalar of a *fixed* candidate explanation is not point-identified	our unidentified object is a latent *annotation* not determined by the observable law. Different object, different mechanism.
Self-explaining GNN semantics (Azzolin et al., 2025)	what the built-in explanation of a trained self-explaining model *is*	we bound node identification under equally likely targets with a common observation law before choosing a model.
WL-aligned proxy-GT benchmarks (Fontanesi et al., 2026)	use WL colours to *align* a GNN's decision with a discriminative motif so an explainer *can* be scored	we construct graph pairs with identical observable law and *opposite* node ground truth to exhibit *non*-identifiability. The construction goal is reversed.
MIL / graph correspondence (Dietterich et al., 1997; Ilse et al., 2018; Tu et al., 2019)	graph = bag, node = instance as a modelling device	we characterize the node assignments compatible with graph-level supervision through their observation laws.
Coverage-equivalent statements in fault localization (Baudry et al., 2006; Steimann et al., 2013; Xie et al., 2013; Gonzalez-Sanchez et al., 2011)	statements with identical coverage rows form an indivisible unit and cannot be separated by a spectrum metric. Empirical comparisons are confounded, and locator-independent bounds are called for	we give an estimator-independent Bayes bound and a finite-sample certificate. We also measure collisions that survive observations richer than the coverage matrix.
Fault equivalence in the classical digital-test sense and diagnostic resolution (McCluskey & Clegg, 1971; Tekumalla & Davidson, 2002; Waicukauski & Lindbloom, 1989)	given a circuit and a test set, some faults admit no distinguishing test. The achievable diagnostic resolution can be computed before testing	that analysis is deterministic and per-instance. Ours is a statistical statement about a *learner* receiving graph-level supervision, with a confidence-valid finite-sample bound and one definition spanning circuits, programs and molecular graphs.

C AMBIGUITY-CLASS CARDINALITY AND FINITE-GRAPH EVALUATION

This appendix supports Section 4.2 with the readout characterization and finite-graph evaluation of Lemma 8.

The ambiguity class contains the assignments that preserve every within-colour relevance count.

$$\mathcal{A}(\ell) = \Big\{ \ell' : \textstyle\sum_{v \in C} \ell'(v) = \sum_{v \in C} \ell(v) \text{ for every colour class } C \text{ of every } G \Big\}. \tag{9}$$

Let $k_C = \sum_{v \in C} \ell(v)$ denote the number of relevant nodes in class C. The class has cardinality $\prod_{G \in \mathcal{G}} \prod_C \binom{|C|}{k_C}$ because each colour class independently chooses which k_C members are relevant. When $k_C = 0$ or $k_C = |C|$, that colour class contributes only one possible assignment. Each choice preserves the relevant-node count in every colour class and therefore belongs to the ambiguity class in Eq. (9). The statistical bound and this ranking characterization describe the population limit, with its finite-sample counterpart developed in Section 4.3.

Anonymous message passing. Part (i) uses only that the message and update maps are shared across nodes, that AGG_t is a function of the neighbour multiset, and that the graph readout is permutation-invariant. This is the anonymous MPNN class of (Xu et al., 2019). Nodes carry no identity beyond their type. The reduction fails precisely when this anonymity condition is violated. Unique IDs and random features can violate it. A positional encoding that distinguishes automorphic nodes also changes the original representation class. When the message and aggregation maps and the update maps are continuous, they induce continuous colour maps f_t on the finite reachable colour set. Every map on this finite set is measurable.

Sufficiency of the colour histogram. For a decomposable readout the graph logit and its graph-level loss are functions of $(n_c(G))_c$ by (6). Because latent relevance is absent from the training observations, replacing it by another member of (9) leaves the optimization problem unchanged. This is the training-time counterpart of Corollary 7.

Finite-graph enumeration. The enumeration covers all unlabelled simple graphs with all binary feature assignments for $n \leq 5$, and all $32{,}768$ topologies at $n = 6$ with a complementary feature pair selected by a fixed hash rule, under three integer parameter settings. This gives $99{,}402$ graph–feature cases and $298{,}206$ parameter checks. Across $65{,}526$ nontrivial colour-class world pairs the maximum within-colour representation spread, score spread and AUC deviation from $1/2$ are all 0. The enumeration separately records $63{,}304$ even-class complementary pairs and $2{,}222$ odd-class equal-count pairs, which avoids the impossible odd-class combination of a complementary swap at equal count. Unique node identities and node-private weights both trigger the assumption-breaking controls.

D FULL CERTIFICATE EVALUATION

This appendix supports Sections 4.3 and 5.4.2 with the certificate computation and its evaluation on known observation laws.

D.1 CERTIFICATE COMPUTATION AND SAMPLING DESIGN

This section supports Section 4.3 by explaining how conditional observation counts determine the bound.

The identity defining $A(P)$ gives optimal accuracy. The inequality $A(P) \leq B(P)$ is tight for two targets and for observation-equivalent channels, where both equal $1/m$. Appendix A.9 proves the main bound V_n, while Appendix A.8 transfers TV errors through the mixture to prove the comparator U_n.

Both use the same radius $r(m, K, n, \alpha)$. The Hoeffding and union-bound proof in Appendix A.8 is an event-counting derivation of the finite-alphabet TV inequality used in the main text: there are $2^K - 2$ nonempty proper events for each of m targets. Here K is the complete specified alphabet, n is the sample count per target, and α controls their simultaneous error probability.

Sampling design. We use candidate counts $m \in \{2, 4, 8\}$ and samples per state $n \in \{100, 500, 2000\}$ with alphabet size 4 and $\alpha = 0.05$. The 10,000 trials cycle through the nine (m, n) combinations. The additional trial uses $m = 2$ and $n = 100$. Each trial draws a shared base $b \sim \mathrm{Dirichlet}(2, 2, 2, 2)$ and a shared perturbation magnitude $s \sim \mathrm{Uniform}(0, 0.35)$. For target z indexed from zero we draw $j_z \sim \mathrm{Dirichlet}(3, 3, 3, 3)$ independently. We normalize $(1 - s)b + 0.7se_{z \bmod 4} + 0.3sj_z$ to obtain P_z, where e_i is the unit vector at coordinate i. Conditional on these laws, each target supplies an independent multinomial count vector with parameters (n, P_z). The laws share b and s across targets.

Computing the main bound and comparator. Given the $m \times K$ table of target-conditioned counts, normalize each row by n to obtain $\widehat{P}_z$. The main computation takes the maximum entry in each observation column, sums these maxima and divides by m, yielding $A(\widehat{P})$. Then return $V_n = \min\{1, A(\widehat{P}) + r\}$.

For the TV comparator, first average the rows to obtain $\widehat{\overline{P}}$. Compute each row's TV distance as $\frac{1}{2}\sum_o |\widehat{P}_z(o) - \widehat{\overline{P}}(o)|$, giving $B(\widehat{P}) = (1 + \sum_z \mathrm{TV}(\widehat{P}_z, \widehat{\overline{P}}))/m$. Return $U_n = \min\{1, B(\widehat{P}) + 2r\}$.

Normalization and each bound evaluation take $O(mK)$ operations once counts are available. Collecting those counts requires n observations under each of m targets. Identical empirical rows yield $V_n = \min\{1, 1/m + r\}$ and $U_n = \min\{1, 1/m + 2r\}$. For example, $m = 2$, $K = 4$, $n = 100$ and $\alpha = 0.05$ give approximately 0.68 and 0.86, respectively. These are analytic equal-frequency examples, separate from the sampled-channel averages.

Observation alphabet and evaluation scope. For a fixed finite alphabet the radius scales as $\sqrt{K/n}$. A binary coverage vector over the median 42 Codeflaws tests has a possible alphabet of size 2^{42}, making this finite-sample bound uninformative at the evaluated sample counts. Compressing that vector produces a bound for the compressed observation channel. Accordingly, the real-corpus experiments measure representation-level score equality, while sampled finite channels evaluate V_n and U_n. The shared-model assumptions remain those of Lemma 2.

Table 7: Supplementary population and TV-comparator checks. Finite-sample rows use U_n ($\alpha = 0.05$, $K = 4$, $m \in \{2, 4, 8\}$ and 10,000 total trials). Their coverage event is $U_n \geq B(P)$ and their slack is $U_n - B(P)$. The main comparison against optimal accuracy $A(P)$ is Table 5.

quantity	value
enumerated discrete channels	9,324
bound violations	0
max(Bayes − bound)	0
finite-sample coverage, n=100 / 500 / 2,000	1.0 / 1.0 / 1.0
median slack, n=100 / 500 / 2,000	0.394 / 0.173 / 0.086
90th-percentile slack, n=2,000	0.093
target-independent Markov controls	1,000
max Bayes-accuracy change	3.33×10^{-16}
equivalence gap, $m \leq 32$	0
target-aligned release rate	1.0
independent population / finite channels	12,000 / 12,000 (0 violations)

Table 7 evaluates the population relaxation and its finite-count comparator. Its $U_n - B(P)$ reference excludes the relaxation gap $B(P) - A(P)$, so it differs from both excess columns in Table 5. The three sample sizes contain 3,334, 3,333 and 3,333 trials. Independent target-free Markov transformations preserve Bayes accuracy up to numerical precision. Equivalent channels have accuracy $1/m$, while the target-aligned channels in this construction attain accuracy one. These checks evaluate the specified observation laws.

The population relaxation agrees with $A(P)$ up to rounding for the sampled $m = 2$ and $m = 4$ channels. At $m = 8$, its gap $B(P) - A(P)$ has median 0.043, mean 0.044 and maximum 0.094. This additional relaxation gap helps explain why U_n can remain looser as the sampling radius shrinks.

E TWO-WORLD BENCHMARKS AND CONTROLLED INSTANTIATIONS

This appendix supports Sections 4.2 and 5.4.1 with graph constructions and comparisons of target information.

Construction. Each graph is assembled from abstract primitives. An anchor node carries an asymmetric context tree of 2–5 irrelevant nodes, providing colour diversity. Attached identically to the anchor is a symmetric motif pair comprising two copies of the same motif (a triangle or a square) each with a designated witness slot. The two witness slots (w_1, w_2) are automorphic, hence share a 1-WL colour at every iteration. The graph label is the motif type, a function of the observable graph alone, satisfying (W1). World A marks w_1 relevant, world B marks w_2. The swap is within a colour class and satisfies (W2). It flips at least the two twin nodes and satisfies (W3).

Graph population. The benchmark uses random state 20260718 and $k = 3$ WL iterations. It has 300 graphs and a global colour vocabulary of size 28. Node counts range over 9–14 with mean 11.353. The positive graph-label fraction is 0.56 and the worlds differ on at least 2 nodes in every graph. Direct evaluation confirms world-independent graph labels $(300/300)$ and within-colour relevance swaps $(300/300)$. The worlds differ $(300/300)$. An individualized per-node table also differs across worlds $(300/300)$, confirming distinct node targets. The global colour-by-relevance contingency table over the 28 colours is byte-identical between worlds, with maximum absolute difference 0.

Executable semantic instantiations. A seven-node Boolean circuit contains two isomorphic branches with a shared logic operator. One branch operator is flipped in the latent world, and the graph label states whether the corrupted circuit ever disagrees with the healthy circuit over all 2^4 input assignments. A compiled Python function applies the same operation to two equal-width input blocks and combines their outputs, with a one-unit mutation inserted after either branch call. Exhaustive binary inputs certify whether the executed program differs from its clean version, and arities $2, 3, 4$ yield AST graphs with $7, 9, 11$ nodes. Operation identities and coarse AST roles are visible. Slice identifiers and the mutation site are withheld. Both domains carry an optional private anchor placed only on the faulty node. Both anchor conditions in this base suite use graph-label loss alone.

Results. Table 8 aggregates 100 base runs from the 188 result files carrying the complete architecture schema. The remainder are arity, held-out-operation, explanation and depth/width checks. Every domain–architecture group attains graph accuracy 1.0. Without a private anchor, within-orbit AUC ranges from 0.489 to 0.520 and the maximum mean mirror-score gap is below 10^{-3}. With the anchor, within-orbit AUC is 1.0 in every base group. The separation repeats for mean-aggregation MPNN and normalized GCN and GIN. It holds for all three program sizes on both devices. Across 20 no-anchor cells the mean explanation AUC is 0.501 for input gradients and 0.500 for feature occlusion. The pooled-node mean is 0.501. Every value is within 0.021 of chance. In held-out-operation tests the no-anchor barrier persists even when two of twelve graph generalization cells fall below 0.90, while all anchored cells retain graph accuracy and node AUC 1.0. Depth/width sensitivity shows the converse boundary, since two underconstrained settings fail to exploit an anchor that is present.

Table 8: Executable symmetry benchmark. The AUC range is over all no-anchor base runs. Anchor AUC is constant over the corresponding runs.

domain	architectures	base runs	graph acc.	no-anchor / anchor AUC
Boolean circuit	3	26	1.0	$[0.489, 0.520]$ / 1.0
Python program AST	3	74	1.0	$[0.492, 0.511]$ / 1.0

Modern architectures and EPFL mutations. The architecture grid covers GAT (Veličković et al., 2018) and GPS (Rampášek et al., 2022) alongside a graph Transformer (Vaswani et al., 2017). It also includes GraphSAGE and the official PyG PNA implementation (Fey & Lenssen, 2019; Corso et al., 2020). We evaluate each on both executable domains with three seeds per cell. All 30 primary runs attain graph accuracy 1.0 while the ten architecture–domain group means for within-orbit AUC lie in $[0.494, 0.512]$. Eight independently coded explanation rules contribute 128 valid runs at chance, and six balanced PyG GNNExplainer runs have mean AUC 0.490. Target-independent spectral projectors

Table 9: EPFL within-orbit AUC by architecture and split. Columns 00, 01, 10 and 11 give (anchor, node supervision) means over the three original seeds. The last column combines those three with three additional anchor-only seeds, reporting six-run mean [minimum, maximum].

Model	Split	00	01	10	11	Expanded 10
GIN	module	0.498	0.498	0.961	1.000	0.863 [0.392, 0.980]
GIN	random	0.522	0.515	0.980	1.000	0.986 [0.951, 0.999]
GraphSAGE	module	0.475	0.538	0.980	1.000	0.977 [0.941, 1.000]
GraphSAGE	random	0.513	0.495	0.999	1.000	0.999 [0.996, 1.000]

and unique identifiers also remain at chance, while the target-aligned anchor plus node supervision gives graph accuracy and orbit AUC 1.0 in all 24 escape runs. The EPFL suites (Amarú et al., 2015) replace generated templates with seven unmodified combinational designs into which $1{,}200$ unique component faults are injected and executed against the clean circuit. Module-held-out splits prevent the conclusion from depending on memorizing one source design. Two suites are run on this material and are reported separately here. The *regime suite* crosses GIN/GraphSAGE with random/module splits. Its graph-only and node-supervision-only settings are compared with target escape. Table 10 summarizes the resulting twelve groups. Its target-escape setting supplies both the anchor and node-level supervision, which is why those four groups sit at 1.0. The *side-information factorial* uses the same architecture and split grid as a 2×2 over (target anchor, node supervision) and is the suite that separates the two ingredients. We average the factorial results across GIN/GraphSAGE and random/module splits. Table 9 preserves the original factorial and the expanded anchor-only batch separately. Anchor-only variability is concentrated in GIN under module holdout. The low run still predicts graph labels accurately, showing that graph-task success can coexist with incomplete use of a supplied target feature.

Table 10: Architecture and external-design boundary, from the EPFL regime suite. "No target" includes graph-only and node-supervision-only cells. The target escape setting supplies the one-bit fault anchor *and* node-level supervision together. The side-information factorial isolates the anchor-only setting (Table 4). Values are group means over three seeds.

suite	families	groups	graph qualification	orbit AUC
controlled, no target	5	10	acc. 1.0	0.494–0.512
EPFL, no target	2	8	mean acc. ≥ 0.937	≤ 0.538
EPFL, target anchor	2	4	mean acc. ≥ 0.969	1.0

Trained complete-colour stress test. Each complete graph has 16 red/blue nodes, with red count uniform on $1,\ldots,15$ and graph label $\mathbf{1}\{n_{\mathrm{red}} \geq 8\}$. Worlds A and B have identical inputs and graph labels and complementary hidden assignments inside the red colour class. A two-layer GCN and a two-layer DeepSets model are trained for 80 epochs on $4{,}096$ graphs and evaluated on $1{,}024$ disjoint graphs. For every seed $66/67/68$, both models attain graph AUC, accuracy and colour-presence AUC 1.0 with world-A/world-B within-red AUC $0.5/0.5$. The maximum over same-colour score spread and node-permutation logit error is 1.42×10^{-14} for the GCN and 2.84×10^{-14} for DeepSets. The run with random state 66 gives the same values for all five metrics on the other device.

WL-histogram and representation comparisons. Across eight independent runs, logistic regression on the WL colour histogram attains graph accuracy 1.0. Its attribution AUC is 0.449 in both worlds with pair AUC 0.5. The between-world AUC difference is 0 with interval $[0, 0]$. The shared witness colour receives a slightly below-average weight. A two-layer symmetric-normalized GCN with fixed random weights and pooled linear readout gives graph accuracy 0.729. Its automorphic-pair embedding gap is 1.38×10^{-16} with interval $[1.05, 1.70] \times 10^{-16}$. The AUC difference is -0.00065 with 95% interval $[-0.007, 0.006]$. Adding a within-copy identifier and training on node labels gives in-world node AUC and pair AUC 1.0. Across worlds the pair AUC is 0.0 and the node AUC is 0.756. The separation gap is 0.244 with interval $[0.226, 0.263]$. With node supervision alone the in-world attribution AUC is 0.952. Pair AUC remains 0.5 both within and across worlds, with separation gap 0 and interval $[0, 0]$. Node supervision identifies the relevant colour class while the identifier distinguishes its members.

MNIST-component boundary control. A MNIST-component construction (Lecun et al., 1998) uses two identical digit components to define the hidden worlds. Within-component AUC is $0.5/0.5$. Graph AUC ranges from 0.692 to 0.731 and accuracy from 0.644 to 0.702. The presence-control AUC ranges from 0.683 to 0.706. Neither architecture reaches the observable-task threshold. We use this construction as a boundary control because tied scores alone do not establish separation from successful graph prediction.

F Approximate symmetry and the escape price

This appendix supports Section 5.3 by examining approximate symmetry and the information supplied by target-dependent queries.

F.1 Representation changes and label mechanisms

This section supports the observation comparisons in Section 5.3.

We evaluate this distinction by appending target-independent random coordinates to the graph inputs. Table 3 compares held-out pair AUC with the original-input result, and Table 27 gives the within-pair score differences. For the corresponding GraphSAGE-only Defects4J comparison, random coordinates give mean pair AUC 0.68 on the root-held-out split and 0.416 on the project-held-out split. The root-held-out values range from 0.48 to 0.80 across repeated training runs. The original-input comparison and the pair counts for both splits are reported in Section 5.2.1. Relative to the original-input ties, the mean rises on the root-held-out split and falls on the project-held-out split. This comparison measures the effect of arbitrary node distinctions on realized rankings. Theorem 1(iii) characterizes the corresponding population ceiling under conditional independence.

Relevance-dependent graph labels. A target anchor supplies target information directly. Graph labels can also carry relevance information when their generating mechanism depends on relevance. For example, let $\rho(c) \in [0, 1]$ denote a colour-level relevance probability. Consider a noisy-OR mechanism in which the graph label is positive when at least one node independently activates with its colour's probability. This gives

$$P(Y{=}1 \mid G) = 1 - \prod_{v} (1 - \rho(c(v))),$$

where the product is the probability that no node activates. The graph label thus carries information about colour-level relevance probabilities. When candidate worlds change ρ and thereby change this label distribution, their comparison falls outside Assumption 1. Exchanging members within a colour class while preserving ρ leaves the displayed probability unchanged. Section 5.3 compares the resulting observation choices on real software faults.

Target signal and nuisance perturbations. The perturbation grid asks whether raw representation distance predicts correct attribution across distinct mechanisms. It uses 32 cells in two executable domains, with three seeds per cell, a five-step mean MPNN with hidden width 128, and 80 training epochs. Table 11 shows the uncorrupted representative cells. Nuisance features enlarge hidden distances while leaving orbit AUC near chance. A target-aligned coordinate of magnitude 10^{-3} permits separation in both domains. Across the full grid, raw hidden distance has diagnostic AUROC 0.691, below the prespecified 0.80 diagnostic criterion, whereas the signal fraction $s/\max(s + u, 10^{-12})$ gives 0.833, for signal magnitude s and nuisance magnitude u. The diagnostic labels an example positive when its faulty member receives a strictly higher score. Thus distance alone does not identify whether a change carries useful target information.

Target-dependent queries. The query construction asks how many response bits can distinguish members of a finite ambiguity class. We evaluate class sizes $2, 3, 4, 5, 7, 8, 16, 32$. Table 12 contrasts individual tests with binary codes and records the noisy-code control. These are constructions quantifying information acquisition cost. Their responses depend on the target and therefore alter the observation law.

Table 11: Representative noiseless, unperturbed-edge cells from the 32-cell, 96-run perturbation grid. Each row is a three-seed mean. Signal is target-aligned feature magnitude; nuisance is target-independent feature magnitude. Hidden distance uses the representation's raw units.

Domain	Signal	Nuisance	Orbit AUC	Hidden distance
Circuit	0	0	0.501	4.93e-06
Circuit	0	0.001	0.498	0.00891
Circuit	0	0.01	0.495	0.089
Circuit	0	0.05	0.500	0.224
Circuit	0	0.1	0.504	0.265
Circuit	0.001	0	1.000	0.022
Circuit	0.01	0.01	0.754	0.356
Python AST	0	0	0.498	4.55e-05
Python AST	0	0.001	0.496	0.038
Python AST	0	0.01	0.507	0.207
Python AST	0	0.05	0.505	0.953
Python AST	0	0.1	0.494	1.89
Python AST	0.001	0	1.000	0.035
Python AST	0.01	0.01	0.758	0.777

Table 12: Queries supplying target-dependent information. Noiseless binary codes distinguish candidate indices. The repeated-code accuracy is the minimum measured across the noisy-code evaluation, at independent 5% response-bit noise.

Construction	Candidates / noise	Queries or accuracy
Individual one-versus-rest probes	32	31 queries
Binary index probes	32	5 queries
Query reduction, same 32 candidates	0% noise	83.9%
Repeated response codes	5% noise	0.965 accuracy

G Classic graph datasets as boundary controls

This appendix supports Section 5.4.3 by comparing natural relevance masks with exchanged targets under fixed observations.

G.1 Natural targets and observation-dependent comparisons

This section supports Section 5.4 by describing the target definitions and their measured localization outcomes.

Natural explanation labels can use differences between colour classes. Table 14 separates the original random splits, additional seeds and structure-held-out batches. The latter place all graphs sharing a visible atom/edge WL-colour description into one split. Mutagenicity has 1,763 such groups and Fluoride-Carbonyl has 8,488, with zero cross-split group overlap. The Mutagenicity random comparison includes 12 original runs plus 24 additional runs, while its structure comparison has nine runs. The Fluoride-Carbonyl random and structure batches each contain 12 runs. Their natural-mask results remain high in the best-performing combinations after structure holdout.

The best Mutagenicity random-split mean is GAT with integrated gradients. Its architecture-mean graph AUROCs range from 0.994 to 0.999. For Fluoride-Carbonyl the corresponding range is 0.937–0.986 across all four families. The exchanged-pair quantity remains one half because its constructed target assignments preserve the complete observations. It answers a different identification question from recovery of a natural mask.

The Benzene and Alkane-Carbonyl structure batches contain 24 runs in total. The additional host check trains GIN for three seeds on each of BA-2Motifs and MUTAG, with five explanation methods per run. These 30 exchanged-target values remain one half. MUTAG remains an optimization control with unstable graph/explanation training. The cross-host comparison reproduces the constructed equality while retaining this task-specific limitation.

Table 13: Graphs with at least one WL colour class containing both relevant and irrelevant nodes. Denominators are the complete evaluated graph populations; the target column specifies informative-mask counts where applicable. Empty masks cannot create mixed classes. Molecular and motif counts are also zero under stable refinement.

Dataset	Graphs	Natural target	Depth	Mixed graphs
Sentiment trees	9129	Sentiment relevance	2	13
Benzene	12000	Functional mask (6,001 positive)	4	0
Alkane-Carbonyl	4326	Functional mask (375 positive)	4	0
Fluoride-Carbonyl	8671	Functional mask (1,527 positive)	4	0
MUTAG	188	Semantic mask (188 nontrivial)	4	0
Mutagenicity	1768	Semantic mask (640 nontrivial)	4	0
BA-2Motifs	1000	Appended motif	4	0

Table 14: Natural-mask evaluations and separate target-exchange controls. The main four-family grid is GAT, GCN, GIN and GraphSAGE (SAGE). The Mutagenicity structure split uses GAT, GIN and SAGE. Families / seeds identifies repetitions within each distinct batch. Natural AUC is the best architecture–attribution mean unless a particular scorer is named. IG denotes integrated gradients and GE denotes GNNExplainer. All exchanged-pair entries evaluate constructed assignments under fixed observations.

Dataset	Split / batch	Families / seeds	Natural-mask result	Exchanged
Mutagenicity	Random	4 / 3	0.961 (GAT, IG)	0.500
Mutagenicity	Structure	3 / 3	0.957 (SAGE, native)	0.500
Mutagenicity	Random repeats	GIN / 6	0.887 native, 0.709 GE	0.500
Mutagenicity	Random repeats	GAT / 6	0.928 native, 0.735 GE	0.500
Mutagenicity	Random repeats	GCN / 6	0.854 native, 0.549 GE	0.500
Mutagenicity	Random repeats	SAGE / 6	0.956 native, 0.519 GE	0.500
Fluoride-Carbonyl	Random	4 / 3	0.832 (GCN, occlusion)	0.500
Fluoride-Carbonyl	Structure	4 / 3	0.804 (GAT, occlusion)	0.500
Benzene	Random	4 / 3	0.937 (SAGE, IG)	0.500
Benzene	Structure	4 / 3	0.934 (GIN, native)	0.500
Alkane-Carbonyl	Random	4 / 3	0.877 (GAT, IG)	0.500
Alkane-Carbonyl	Structure	4 / 3	0.833 (GAT, IG)	0.500
BA-2Motifs	Structure, second host	GIN / 3	Graph AUROC 1.000	0.500
MUTAG	Structure, second host	GIN / 3	Unstable optimization	0.500

H TRAINING AND MODEL SELECTION

This appendix gives the training configurations used for the real-corpus results in Tables 2 and 3. All models have two message-passing layers with hidden width 64 and dropout 0.1. We use AdamW with learning rate 0.003 and weight decay 10^{-4}. The maximum training length is 40 epochs and the gradient-norm limit is 5. Every architecture and input setting uses the three random seeds $47301/47302/47303$. Tables 2 and 3 use sample standard deviations with denominator $3 - 1$.

Defects4J aggregation and optimization. GraphSAGE averages statement messages at test nodes and test messages at statement nodes. Separate self and neighbour linear maps are followed by ReLU. GAT starts from test-wise means and uses single additive attention with neighbourhood softmax and test-to-node context. The Transformer uses four-head scaled dot-product attention with residual updates and LayerNorm. We optimize weighted binary cross-entropy with logits over valid test outcomes and use log-mean-exp pooling. The positive-class weight is obtained from training-test counts. Each forward pass processes one fault and each optimizer step accumulates eight fault losses. The final shorter group is also divided by eight. GraphSAGE was assigned an RTX3090 and GAT and Transformer were assigned an RTX4090. Saved model records identify CUDA execution.

Codeflaws aggregation and optimization. GraphSAGE concatenates the node embedding with the mean neighbour embedding before a linear map and ReLU. GAT uses four additive-attention heads on concatenated endpoints with LeakyReLU slope 0.2 and per-destination softmax. The Transformer uses four scaled dot-product attention heads with per-destination softmax and no LayerNorm. We optimize test-outcome weighted binary cross-entropy using noisy-OR pooling. The positive-class weight is the training negative-to-positive ratio clipped to $[0.1, 10.0]$. The batch size is 32 faults

Table 15: Evaluation measures and their populations. AUC uses one half for ties. Chance AUC is 0.5; top-1 chance depends on the candidate set.

Measure	Object and aggregation	Interpretation
Test-outcome AUROC	Tests pooled in the specified held-out fault split	Higher is better
Pooled node AUROC	Repaired versus non-repaired candidates, including cross-fault pairs	Higher is better
Macro node AUROC	Mean of within-graph node AUROCs over evaluable graphs	Higher is better
Pair / within-orbit AUC	Specified target/non-target pairs; controlled orbits contain interchangeable candidates	Higher is better
Top-1 success	First ranked node is any repaired statement; averaged over faults	Uniform class: k/m
First repaired rank	Position of first repaired statement in a uniform ordering of the selected class	Lower is better
Maximum score gap	Largest absolute raw-score difference in a matched pair	Zero means equality

Table 16: **Validation test prediction selects Defects4J models before the epoch limit.** Best epochs are indexed from zero. Validation AUROC is higher-is-better (↑).

model	seed	epochs run	best epoch	validation test AUROC ↑
GraphSAGE	47301	23	15	0.977
GraphSAGE	47302	14	6	0.970
GraphSAGE	47303	19	11	0.968
GAT	47301	22	14	0.973
GAT	47302	12	4	0.967
GAT	47303	22	14	0.976
Transformer	47301	18	10	0.990
Transformer	47302	21	13	0.989
Transformer	47303	24	16	0.985

and training uses CUDA. Attribution evaluation uses CPU float64 in batches of 16 faults, with at most eight CPU threads. Coverage is collected with `gcc -coverage` and `gcov`. The difference between faulty and fixed source lines supplies repaired-node truth. Candidate pairs share coverage and descriptors, with WL refinement checked separately.

Validation and selected models. We select models using pooled validation test-outcome AUROC and restore the best parameters. Both datasets use early-stopping patience 7. A Defects4J improvement must exceed 10^{-5}, while Codeflaws requires any strict increase. The 33 Defects4J validation faults contain 111 matched pairs. Their test outcomes determine model selection, while repaired-node labels are used for localization evaluation. Tables 16 and 17 report the actual training lengths and selected validation scores.

I DEFECTS4J OBSERVATIONS

This appendix supports Section 5.2.1 with the Java fault population and the construction of its observations.

Data and observations. We use recorded faults with repaired statements as node targets because their test outcomes and repairs distinguish prediction from localization. We study Java faults from the public GZoltar coverage archives for Defects4J (Just et al., 2014; Campos et al., 2012) and C faults from Codeflaws (Tan et al., 2017). The change in language and test collection lets us examine the same observation choice in different software workloads. We form test–statement bipartite graphs from coverage and enrich statement features with spectrum statistics and local program descriptors. For Defects4J these descriptors encode abstract syntax tree (AST) statement type and depth as well as bucketed method size and local control-flow degree. For Codeflaws the statement type is the leading keyword of the line and the remaining fields use the line-level successor graph inside the enclosing function. They add syntax and structure to coverage without supplying source identifiers or

Table 17: **Codeflaws model selection uses validation test outcomes across input settings.** Validation AUROC is higher-is-better (↑). The anchor provides oracle target information.

model	input	seed	epochs run	validation test AUROC ↑
GraphSAGE	original	47301	40	0.895
GraphSAGE	original	47302	40	0.897
GraphSAGE	original	47303	40	0.894
GAT	original	47301	40	0.901
GAT	original	47302	40	0.896
GAT	original	47303	40	0.899
Transformer	original	47301	40	0.896
Transformer	original	47302	40	0.894
Transformer	original	47303	40	0.896
GraphSAGE	target anchor	47301	40	0.897
GraphSAGE	random coordinates	47301	32	0.892
GraphSAGE	target anchor	47302	40	0.894
GraphSAGE	random coordinates	47302	40	0.893
GraphSAGE	target anchor	47303	40	0.895
GraphSAGE	random coordinates	47303	40	0.894

Table 18: Defects4J native scores and GNNExplainer masks on the same 31 root-held-out faults (4,279 candidates). One trained GraphSAGE model, seed 47301; one explainer evaluation. Interval and permutation operate on faults, not independent pairs.

Scorer	Node AUROC	Pair AUC	Exact ties / 25	Max gap
Native GraphSAGE	0.623	0.500	25	0
GNNExplainer	0.566	0.360	0	0.127

the repaired location. We evaluate repaired/non-repaired pairs with identical coverage and descriptors that remain indistinguishable under feature-preserving colour refinement.

GNNExplainer configuration. We evaluate GNNExplainer (Ying et al., 2019) on all 31 root-held-out graphs using the GraphSAGE model trained with seed 47301. Each statement receives one soft mask multiplying all its input features. We optimize node-object feature masks with edge masking disabled in the dense coverage model. The explanation target is the original model's binary prediction on each valid test, with a positive logit predicting failure. Repaired-node labels are excluded from mask optimization. We use random initialization with seed 20260921 and optimize for 100 epochs at learning rate 0.01. Mask optimization uses CPU float32 with four threads and the installed PyG default sparsity and entropy coefficients. The masked and original predictions agree on 98.3% of tests. Table 18 compares the native float64 forward pass with this single mask optimization on the same model and candidate set. Fault resampling and a permutation that exchanges outcomes jointly within each fault account for the 25 pairs clustering in 15 faults (at most four pairs per fault). The observed pair accuracy supplies no evidence of improvement over chance. Independently optimized masks permit unequal scores, while their exact source of separation is not isolated by this one run.

Comparable spectrum baselines. Table 21 evaluates the same fault splits and candidate nodes as the neural comparison in Table 2. Pooled node AUROC combines nodes across faults, while macro node AUROC averages the separate fault-level AUROCs. The full-pool Ochiai result in Table 22 uses a different candidate population and is not directly comparable to either held-out column.

I.1 Corpus populations and neural splits

This section supports the population comparisons in Sections 5.2.1 and 5.2.2.

To measure how often this ambiguity survives richer observations, we compare coverage-only matching with matching after adding local descriptors. Table 23 reports the corresponding fault counts and separates them from the neural evaluation splits. Project-held-out pair AUC is 0.500 ± 0.0000 for each model in Table 2. The nearby-statement relation requires the same top-level source class and a line-number difference of at most two. Of the matched faults, 95 include a pair outside this relation. This source-location check describes match provenance; source locations remain withheld from model inputs. The reduction from coverage-only matches follows from the additional syntax and

Table 19: Uncertainty and signed score differences for the single GNNExplainer evaluation. Correct order occurs in 9 of 25 pairs; 16 signed differences are negative.

Fault-level diagnostic	Value
Pair-AUC bootstrap 95% interval	$[0.190, 0.500]$
Joint within-fault permutation p	0.119
Mean repaired-minus-non-repaired score $\pm$ sample SD	-0.009 ± 0.029
Fault-bootstrap interval for mean difference	$[-0.024, -0.001]$

Table 20: Defects4J random-coordinate runs. Pair AUC is measured on 25 root-held-out pairs, 231 project-held-out pairs and 111 validation pairs. Each row is one training run. The nine-run project mean is 0.467 (sample SD 0.116); the GraphSAGE-only mean is 0.416. Root-held-out AUC varies in steps of 0.04 because it contains 25 pairs.

Model	Seed	Root held out	Project held out	Validation
GAT	47301	0.640	0.424	0.378
GAT	47302	0.480	0.407	0.360
GAT	47303	0.320	0.710	0.514
GraphSAGE	47301	0.800	0.550	0.225
GraphSAGE	47302	0.760	0.364	0.423
GraphSAGE	47303	0.480	0.333	0.369
Transformer	47301	0.440	0.476	0.441
Transformer	47302	0.480	0.528	0.414
Transformer	47303	0.240	0.407	0.649

structural distinctions available to the scorer. The remaining matches establish that local descriptors enrich the observation without resolving every candidate target.

Defects4J provides faulty and fixed Java program versions together with tests that expose each fault (Just et al., 2014). For each fault we form a test–statement bipartite graph. Statement features contain spectrum statistics and AST type. They also contain a method-size bucket and relative AST depth alongside local control-flow degree buckets. Paths and line numbers are withheld, as are source digests and patch identity. A fault enters the study when its coverage archive parses, its failing tests cover the patched statement, and at least one non-patched candidate remains after filtering. Here 330 of 395 archives qualify. Root-held-out splits share no fault root, and the project-held-out split withholds Time. The pair statistic counts ties as one half, so a model that assigns two statements identical scores contributes one half to it.

Localization under class-constant rankings. The oracle selects a tied class maximizing its repaired fraction and orders that class uniformly. For class size m with k repaired statements, top-1 success is k/m and expected first repaired rank is $(m + 1)/(k + 1)$. Table 24 averages these quantities over faults. The restriction to a uniform within-class ordering is part of this ceiling.

Multiple repaired statements can make one target accessible even when another is tied. Of 115 matched Defects4J faults, 64 also contain an unmatched repaired statement, keeping this optimistic success at one. Restricting to single-repair faults exposes the greater within-class loss. Codeflaws has one repaired line per eligible fault. Its matched classes have size two in 407 of 830 cases and reach size 55.

J CODEFLAWS OBSERVATIONS

This appendix supports Section 5.2.2 with the C fault population and the construction of its observations. The C0 observation uses coverage alone and C1 adds each statement line's leading-keyword descriptor. Defects4J uses AST and local structural descriptors instead (Appendix I). These labels distinguish the observation sets compared below.

Table 26 separates population selection from the effect of adding tests. On the common population, additional tests remove some descriptor matches while leaving roughly one third of faults matched. Table 31 gives the full observation-count breakdown.

Table 21: **Matching the candidate population changes spectrum-method ranking.** All entries are deterministic evaluations on the neural candidate sets. Ranking metrics are higher-is-better (↑). Each root-held-out row contains 31 faults and 4,279 statements, including 75 repaired statements. Each project-held-out row contains 19 faults and 4,342 statements, including 85 repaired statements. Every pair is an exact tie with zero score gap.

split	method	pooled node AUROC ↑	macro node AUROC ↑	pair AUC ↑	pairs
root held out	Ochiai	0.504	0.626	0.500	25
root held out	DStar	0.498	0.627	0.500	25
root held out	Tarantula	0.553	0.624	0.500	25
project held out	Ochiai	0.487	0.628	0.500	231
project held out	DStar	0.461	0.628	0.500	231
project held out	Tarantula	0.514	0.626	0.500	231

Table 22: **Full-pool coverage-based ranking on Defects4J.** This population is distinct from the neural held-out comparisons in Table 2. Higher node AUC indicates better ranking.

full-pool reference	faults	candidate statements	repaired statements	node AUC ↑
Ochiai	330	499,425	713	0.848

The attribution methods compare native node scores with input sensitivity and the effect of removing node features. Both corpora append eight independent standard Gaussian coordinates per statement and keep them fixed during a run. Defects4J generates each fault's coordinate matrix from a seed determined by the training seed, project and fault identifier. Codeflaws seeds each statement separately using the training seed, fault root and statement index. These protocols provide arbitrary input distinctions without consulting repair labels. The Codeflaws target anchor instead appends the binary repaired-node label as one coordinate.

The score-gap statistic is the maximum across training runs. Defects4J withholds fault roots or the Time project. Table 27 gives non-equivalent-pair metrics and raw score gaps. Score gaps are measured in each scorer's units and are not a scale-invariant measure of target information. Table 30 provides all original-input combinations. Table 28 gives the additional attribution combinations. Table 29 reports the held-out observation comparisons. The Defects4J random-coordinate panel covers GAT, GraphSAGE and the bipartite Transformer at seeds 47301, 47302 and 47303. Its nine project-held-out pair AUCs range from 0.333 to 0.710 with three values above 0.5, which an exact two-sided sign test places at $p = 0.508$. The per-architecture project-held-out means are 0.514 for GAT, 0.416 for GraphSAGE and 0.470 for the bipartite Transformer. Held-out test-outcome AUROC over the same nine runs averages 0.985 and all-node AUROC averages 0.867.

The Codeflaws distribution (Tan et al., 2017) unpacks to 3,904 defect/fix directories mined from programming-contest submissions. The corpus paper reports 3,902 defect/fix pairs, and every count below is taken over the 3,904 directories present on disk. Coverage matrices are produced by a two-stage `gcc -coverage` build followed by `gcov`, so that the per-source coverage artefacts are named after the source file. A defect enters the study when its directory builds and its test suite contains both passing and failing outcomes. The fix must touch a single old line covered by at least one failing test. At least five statements including the repaired one must also be fail-covered. No directory failed the coverage build, so the build condition excludes none. The outcome-balance condition excludes 387 directories and the single-line-fix condition excludes 708. The repaired-line coverage condition then excludes 244 and the five-candidate condition excludes 56. The resulting population contains 2,509 usable faults. The observation channel is the full shipped test suite together with the positive, negative and held-out tests, because the shipped suite alone is often only three tests. Both channels are reported in Table 31, and the full channel supplies the observations used in the main comparisons. The observed values are 2,509 faults and 1,026 problems. There are 830 C1 collision faults at rate 0.331 and 1,986 C1 collision pairs.

Table 23: **Local descriptors leave equivalent statement pairs in both software workloads.** The upper block gives exact corpus counts and the fraction of eligible faults containing a matched pair. The lower block gives the Defects4J neural splits and their evaluated pairs. The Codeflaws input count refers to directories on disk, while the corpus publication reports 3,902 defect/fix pairs. A dash denotes an unreported or inapplicable entry. These are population descriptions rather than estimates averaged across training runs.

population or observation	Defects4J	Codeflaws
input records	395 coverage archives	3,904 directories
eligible faults	330	2,509
source groups	six projects	1,026 problems
coverage-only matched faults	270	2,150
coverage-only matched fraction	–	85.7%
coverage-only pairs	–	20,523
descriptor-matched faults	115	830
descriptor-matched fraction	34.8%	33.1%
descriptor-matched pairs	924	1,986
source groups with descriptor matches	five projects	550 problems
matched faults beyond nearby-statement relations	95	–
median tests per fault	–	42
median candidates covered by failed tests	–	15

Defects4J neural population	total	training	validation	root held out	project held out
faults	255	172	33	31	19
evaluated equivalent pairs	–	–	111	25	231

Table 24: Optimistic performance under uniform ordering within the best tied class. Values are exact averages on the specified finite populations. A successful top-1 prediction may be any repaired statement.

Corpus	Population	Faults	Top-1 ceiling	First repaired rank
Defects4J	All eligible	330	0.899	1.31
Defects4J	Matched	115	0.711	1.90
Defects4J	Single repair	124	0.800	1.72
Defects4J	Single repair, matched	36	0.310	3.47
Codeflaws	All eligible	2509	0.794	1.40
Codeflaws	Matched	830	0.377	2.20

Table 25: **A target anchor improves Codeflaws pair ranking and its accuracy varies across runs.** Entries are mean ± sample standard deviation over three runs. Test and node AUROC use held-out observations. Pair AUC uses all 1,986 matched pairs including training pairs. Higher is better (↑), and the anchor supplies oracle target information.

model	attribution	test AUROC ↑	node AUROC ↑	pair AUC ↑
GAT	native score	0.887 ± 0.0021	0.618 ± 0.0145	0.500 ± 0.0000
GraphSAGE	native score	0.884 ± 0.0012	0.619 ± 0.0118	0.500 ± 0.0000
GraphSAGE	grad×input	0.884 ± 0.0012	0.620 ± 0.0068	0.500 ± 0.0000
GraphSAGE	occlusion	0.884 ± 0.0012	0.622 ± 0.0098	0.500 ± 0.0000
Transformer	native score	0.881 ± 0.0010	0.617 ± 0.0158	0.500 ± 0.0000
GraphSAGE + random coordinates	native score	0.881 ± 0.0015	0.641 ± 0.0071	0.502 ± 0.0126
GraphSAGE + target anchor	native score	0.884 ± 0.0012	0.698 ± 0.0399	0.944 ± 0.0812

Table 26: Codeflaws test suites on their own eligible populations and on the same 2,349 faults. Coverage plus descriptors defines a match. The common-population transition removes 60 matched faults, creates zero, and leaves 750 matched and 1,539 unmatched under both suites.

Population	Suite	Faults	Matched faults	Matched pairs
Own eligible	Full	2509	830 (33.1%)	1986
Own eligible	Shipped	2349	810 (34.5%)	1959
Common eligible	Shipped	2349	810 (34.5%)	1959
Common eligible	Full	2349	750 (31.9%)	1786

Table 27: **Non-equivalent Codeflaws candidates permit ranking beyond matched-pair ties.** The upper panel reports non-equivalent-pair AUC as mean ± sample standard deviation over three runs. Max gap is the largest within-pair difference across runs. Higher AUC is better (↑), and smaller gaps indicate closer ties (↓). The lower panel summarizes original-input comparisons across architectures and attribution methods. The anchor supplies oracle target information.

model	attribution	non-eq. AUC ↑	max gap ↓
GAT	native score	0.554 ± 0.0174	3.55e-15
GraphSAGE	native score	0.543 ± 0.0114	3.55e-15
GraphSAGE	grad×input	0.557 ± 0.0106	1.85e-13
GraphSAGE	occlusion	0.550 ± 0.0114	2.84e-14
Transformer	native score	0.547 ± 0.0221	1.78e-15
GraphSAGE + random coordinates	native score	0.584 ± 0.0055	14.2
GraphSAGE + target anchor	native score	0.728 ± 0.0851	13.9

original-input comparison	AUC ↑
non-equivalent pairs, native-score mean across architectures	0.548
non-equivalent pairs, mean across all attribution combinations	0.554
non-equivalent pairs, range across all attribution combinations	0.543–0.560
equivalent pairs, strict floating-point range	0.500–0.502

Table 28: **Additional Codeflaws attribution scores preserve the original-input pair ties.** AUC entries are means over three runs. Test and node AUROC use held-out observations. Pair AUC uses all 1986 equivalent pairs including training pairs at tolerance 10^{-9}. Non-eq. AUC uses 14156 non-equivalent pairs from the same faults. Higher AUC is better (↑), and smaller gaps indicate closer ties (↓). Max gap is the largest within-pair difference across runs. The anchor supplies oracle target information.

model	attribution	test AUROC ↑	node AUROC ↑	pair AUC ↑	non-eq. AUC ↑	max gap ↓
GAT	grad×input	0.887	0.631	0.500	0.560	1.42e-14
GAT	occlusion	0.887	0.621	0.500	0.560	7.11e-15
Transformer	grad×input	0.881	0.594	0.500	0.560	5.68e-14
Transformer	occlusion	0.881	0.621	0.500	0.554	6.75e-14
GraphSAGE + random coordinates	grad×input	0.881	0.642	0.501	0.586	295
GraphSAGE + random coordinates	occlusion	0.881	0.644	0.503	0.588	54.7
GraphSAGE + target anchor	grad×input	0.884	0.686	0.937	0.786	208
GraphSAGE + target anchor	occlusion	0.884	0.702	0.944	0.732	106

Table 29: **Arbitrary identifiers and oracle information give different localization outcomes.** Neural entries are means over three independent runs, and the three-architecture row averages nine runs. Run ranges are minimum–maximum across runs. Codeflaws held-out entries use the GraphSAGE native score. The lower block gives deterministic spectrum-method scores on Codeflaws. Higher AUC indicates better ranking (↑), and a zero maximum gap indicates exact paired-score equality. The oracle uses complete target labels.

corpus	observation and evaluation	pairs	pair AUC ↑	run range
Defects4J	random coordinates (GraphSAGE), root held out	25	0.680	0.48–0.80
Defects4J	random coordinates (GraphSAGE), project held out	231	0.416	0.33–0.55
Defects4J	random coordinates (three architectures), project held out	231	0.467	0.33–0.71
Defects4J	complete node-truth oracle	–	1.0	–
Codeflaws	random coordinates, held out	282	0.467	–
Codeflaws	target anchor, held out	282	0.960	–

Codeflaws spectrum method	held-out node AUROC ↑	pair AUC ↑	max gap ↓	exact ties
Ochiai	0.572	0.500	0	1,986
DStar	0.643	0.500	0	1,986
Tarantula	0.586	0.500	0	1,986

Table 30: All nine original-input Codeflaws combinations. Means ± sample SD over three seeds. Pooled node AUROC uses held-out faults. Pair columns distinguish all 1,986 pairs (including training) from 282 held-out pairs; tolerance is 10^{-9}. Max gap is the largest raw-score difference over the three runs.

Model	Attribution	Node AUROC	All-pair AUC	Held-out pair AUC	Max gap
GAT	Native	0.618 ± 0.014	0.500 ± 0.000	0.500 ± 0.000	3.6×10^{-15}
GAT	Grad×input	0.631 ± 0.004	0.500 ± 0.000	0.500 ± 0.000	1.4×10^{-14}
GAT	Occlusion	0.621 ± 0.013	0.500 ± 0.000	0.500 ± 0.000	7.1×10^{-15}
GraphSAGE	Native	0.619 ± 0.012	0.500 ± 0.000	0.500 ± 0.000	3.6×10^{-15}
GraphSAGE	Grad×input	0.620 ± 0.007	0.500 ± 0.000	0.500 ± 0.000	1.8×10^{-13}
GraphSAGE	Occlusion	0.622 ± 0.010	0.500 ± 0.000	0.500 ± 0.000	2.8×10^{-14}
Transformer	Native	0.617 ± 0.016	0.500 ± 0.000	0.500 ± 0.000	1.8×10^{-15}
Transformer	Grad×input	0.594 ± 0.041	0.500 ± 0.000	0.500 ± 0.000	5.7×10^{-14}
Transformer	Occlusion	0.621 ± 0.017	0.500 ± 0.000	0.500 ± 0.000	6.8×10^{-14}

Table 31: Codeflaws occupancy under both observation channels. The full channel uses the shipped suite plus the positive, negative and held-out tests. The shipped channel uses only the tests distributed with the corpus, which are often three per defect. Every value is measured before any model is trained.

quantity	full suite	shipped suite
Usable real faults	2,509	2,349
Distinct Codeforces problems	1,026	973
Distinct contests	512	501
Median tests per fault	42	5
Median fail-covered candidates	15	15
C0 collision faults	2,150	2091
C0 occupancy	0.8569	0.8902
C0 collision pairs	20,523	21,911
C1 collision faults	830	810
C1 occupancy	0.3308	0.3448
C1 collision pairs	1,986	1,959
C1 collision problems	550	535
1-WL collision pairs, iteration 1	1,990	1,978
1-WL collision pairs, iteration 2	1,989	1,962
1-WL collision pairs, iteration 3	1,988	1,962
1-WL collision pairs, iteration 4	1,988	1,962
1-WL stable iteration (median / max)	1 / 3	1 / 4