WHEN GRAPH ABLATIONS CHANGE NOTHING: LOCATING WHERE TIME-SERIES ANOMALY DETECTORS ABSORB A GRAPH REPLACEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph-based time-series anomaly detectors are often ablated by replacing their graph with a reversed or random one; when detection is unchanged, the graph is declared irrelevant. Similar ablation outcomes, however, can arise from different internal responses: the replacement can be absorbed at execution, predictive information, utilization by the fitted model, or anomaly scoring. We introduce **layered graph-ablation diagnosis**, an executable protocol with one measurement and one decision rule per layer that distinguishes these responses and names the component to examine next. At the information layer, we identify a **history information retention mechanism**: older states of a replacement neighbor can still predict the current target, and in Gaussian directed m-cycles under the specified forecasting interface a reversed neighbor first becomes predictive at lag $m-1$, as window and lag interventions on trained forecasters confirm. On a pre-registered benchmark of twelve planted absorbers, four cases have ablation contrasts within ± 0.005 yet absorbers at three different layers; blind rules recover 101 of 120 labels, with all misses in two partial-utilization cases, whereas importance readouts such as GraphMask confound the layers. Applied to public detectors, the execution check finds that four of seven implementations ignore a supplied graph when they learn the graph. On a directed cycle, the diagnosis locates GDN's near-chance detection in its source selection and static attention, and replacing both lifts AP to 0.967. On SMD, it uncovers and repairs a prediction deficit hidden behind a null learned-versus-random contrast, a contrast that stays unresolved after the repair, consistent with redundant sources.

1 INTRODUCTION

Time-series anomaly detection identifies departures from normal patterns in equipment and system records (Schmidl et al., 2022). Graph-based detectors model relationships among sensors and flag anomalies through prediction errors or relational changes (Deng & Hooi, 2021; Zhang et al., 2022). To test whether these relationships matter, practitioners replace the graph with a reversed or random one, and random graphs can perform comparably (Bello et al., 2026). Such a null result is usually read as "the graph is irrelevant." Yet similar ablation outcomes can arise from different internal responses: the model may never read the replacement, older history may keep the information, the fitted model may leave it unused, or the score may ignore it.

These responses call for different next steps, and a single endpoint cannot tell them apart. In two of our planted cases the refitted AP contrast is zero to three decimals: in one the detector symmetrizes the supplied graph, so both graphs execute the same edges; in the other the anomalies are bursts that the score ranks near-perfectly with or without the graph. The first calls for intervening on the executed edges, the second for changing the score. The execution case occurs in practice. Among seven audited public implementations, a supplied graph is ignored in the four configurations that learn the graph, and the released GDN configurations of a recent topology benchmark all enable graph learning (Appendix E.1). Prior work separates predictor quality from scoring quality (Garg et al., 2022) and finds that topology, normal validation loss, and detection relate differently across data and models (Bello et al., 2026), but graph-ablation studies still report one endpoint metric. We

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

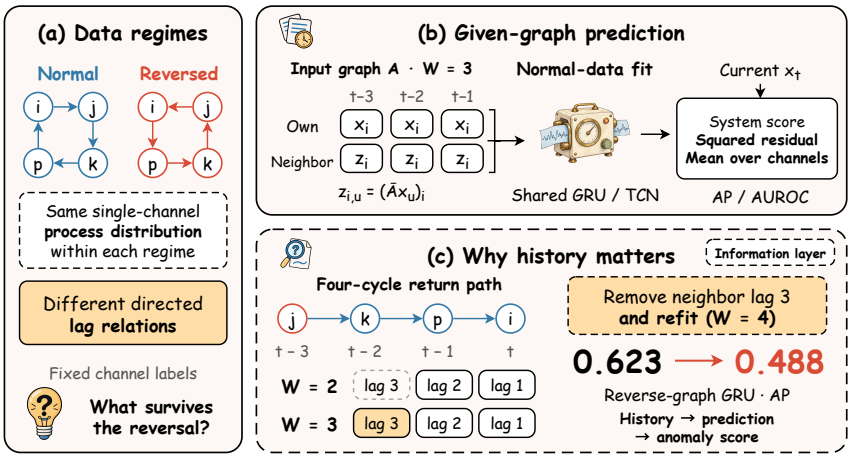


Figure 1: **How reversed-neighbor history retains predictive information**, one way in which the information layer of the diagnosis absorbs a graph replacement (Section 3). (a) Reversing the data regime changes directed lag relations while preserving channel labels and within-regime univariate process distributions. (b) The model reads self-history and aggregated neighbor history under supplied graph A, fits normal data, and scores the cross-channel mean squared residual. Graph selection and the regime change in (a) are separate operations. (c) The normal path $j(t-3) \to k(t-2) \to p(t-1) \to i(t)$ makes reversed-neighbor lag 3 first visible at $W = 3$; k, p are intermediate nodes. At fixed $W = 4$, removing neighbor lag 3 and refitting reduces reversed-graph GRU AP from 0.623 to 0.488 across ten seeds, with common guard $g = 8$.

ask: *when a graph ablation changes nothing, which part of the detector absorbed the replacement, and can changing it move detection?*

We propose **layered graph-ablation diagnosis**. We follow one replacement through four layers, each of which can absorb it: execution (does the model read the new edges?), information (do older history or other sources keep what the edges carried?), utilization (does the fitted model use the available information?), and scoring (does a change in prediction reach the anomaly ranking?). Each layer has a measurement and a decision rule, and each label names a component that we can replace (Algorithm 1). For the information layer, we identify **history information retention**: a reversed neighbor's older states can carry what the removed edges carried, and in a directed m-cycle the reversed neighbor first becomes predictive at exactly lag $m - 1$ (Figure 1).

On a pre-registered benchmark of planted absorbers, four cases whose ablation contrasts all lie within ± 0.005 have absorbers at three different layers, and the blind rules recover 101 of 120 labels; every miss falls in the two cases where a component only partly limits utilization. We then apply the diagnosis where the absorber is unknown (Table 1). On a directed cycle, it locates GDN's near-chance detection at a top-k source selection detached from the loss gradient and at static attention; replacing both raises AP from 0.545 to 0.967. On SMD, behind a null learned-versus-random contrast, GDN's learned graph predicts normal data worse than self-loops; replacing the located components repairs this and lets the graph improve prediction on nine objects, while the contrast stays unresolved.

Our contributions are as follows:

- **Layered graph-ablation diagnosis** (Section 3): four layers, each with a measurement and a decision rule, that distinguish where a replacement is absorbed and which component to change next, released as a toolkit.
- **A computable mechanism and a planted validation** (Sections 3.2–4): history retention is an information-layer absorber with an exact first predictive lag, $m - 1$ in directed m-cycles; on twelve planted absorbers, blind rules separate cases that ablation scores, message importance, and GraphMask confound.
- **Component interventions and real-data applications** (Sections 5.1–6): an execution audit of seven public detectors; located selector and attention replacements that restore GDN's cycle detection; on SMD, a prediction deficit found and repaired behind a null contrast.

Table 1: **Similar endpoints, different diagnoses.** Each row gives the endpoint a graph ablation reports, the measurement the diagnosis adds, the first absorbing layer found by Algorithm 1, and what replacing the located component shows. Controlled rows: directed four-cycle, AP with prevalence 0.50. Real-data rows: AP of the learned graph (GraGOD: topology) minus a degree-matched random graph after refitting; normal-risk ratios compare with self-only support. Native GDN AP is 0.537 in the five-seed execution study and 0.545 in the twenty-seed replacement study. Seeds and objects: Sections 5 and 6.

Endpoint observation	Added measurement	Diagnosis	After intervention
Controlled four-cycle: true versus reversed graph			
GDN, supplied graph: AP 0.537 for both	executed edges and predictions identical	Execution	reversing executed edges inverts the ranking
Graph-GRU (W=3), refit: 0.997 vs 0.628	reversed neighbor predictive from lag 3	Information (retained)	at W=4, removing lag 3 returns reversed AP to chance
GDN, own graph: AP 0.545	parents selected at the random rate; static attention	Utilization	trainable selector and GATv2: 0.967
Real data: learned minus random graph			
GraGOD GDN, SWaT: +0.004	supplied graph ignored; nodes read a mix of sensors	Execution	transposed input, learning off: still unresolved
GDN, SMD (25): −0.005, unresolved	learned graph worse than self-only on all 25	Information (11), utilization (14)	prediction repaired; graph used on 9; contrast unresolved
GDN, PEMS-BAY: +0.005	4.9× self-only normal error	Utilization	root weight: 0.99×, graph used; absorbed at scoring
MTGNN, SWaT: +0.020	1.59× self-only error; depends on channel-max score	Utilization	channel-mean score: AP 0.71 for every support

2 Related Work

Graphs in time-series anomaly detection. Graph detectors learn relations among sensors for forecasting in GDN (Deng & Hooi, 2021), forecasting and reconstruction with attention in MTAD-GAT (Zhao et al., 2020), a Gumbel-softmax graph with a Transformer in GTA (Chen et al., 2022), density estimation over a learned acyclic graph in GANF (Dai & Chen, 2022), reconstruction in FuSAGNet (Han & Woo, 2022), and scoring of relational changes in GReLeN (Zhang et al., 2022). GraGOD compares topologies and finds that random graphs can perform comparably (Bello et al., 2026). NOISEFLIP separates univariate deviations from cross-channel changes with marginal-preserving constructions (Pinet et al., 2026); our cycles, built on cycle and time-reversal analysis (Morales-Mombiela et al., 2013; Winkler et al., 2016), extend this comparison from instantaneous marginals to temporal history. For node- and graph-level learning, structure-learning benchmarks find that learned structures do not consistently outperform the original graph (Zhou et al., 2023; Li et al., 2023). We ask where, inside a detector, a graph replacement stops mattering.

Ablation, attribution, and redundancy. Removing an input and retraining can hide its importance when correlated inputs carry the same information (Hooker et al., 2019). Leave-one-covariate-out inference (Lei et al., 2018), conditional permutation importance (Strobl et al., 2008), and population importance (Williamson et al., 2023) measure an input's value given the other inputs, and Granger analysis asks whether additional history improves prediction (Granger, 1969; Tank et al., 2022). GNNExplainer (Ying et al., 2019) and GraphMask (Schlichtkrull et al., 2021) score the messages of a fixed model. These methods measure how much an input or message matters to one model; the layered diagnosis asks, in addition, at which stage a replacement stops acting, and Section 4 compares both on the same cases. Attribution methods are validated on constructions where the relevant features are planted (Adebayo et al., 2018; Shah et al., 2021); we validate the layered diagnosis in the same way. In multivariate forecasting, channel-independent models often match channel-mixing ones (Nie et al., 2023; Han et al., 2024).

Attention, prediction, and scoring. In GAT-style attention, which GDN uses, the ranking of neighbors does not depend on the target, and GATv2 removes this restriction (Brody et al., 2022). Fixed-predictor loss comparisons (Diebold & Mariano, 1995) and usable information (Xu et al.,

2020) separate available predictive gain from its realized use, and distinct graphs can predict equally well (Prasse & Van Mieghem, 2022). Garg et al. compare predictors and scoring functions separately (Garg et al., 2022); we trace one intervention through both, without point adjustment (Kim et al., 2022).

3 Layered Graph-Ablation Diagnosis

A graph ablation replaces the graph a detector receives and compares a detection metric. The diagnosis takes the detector, its normal and test data, and the replacement. It asks four questions in order, one per layer at which the replacement can be absorbed, and returns every layer's verdict, the first absorbing layer, and the component to examine next (Algorithm 1). A layer *absorbs* a replacement when it limits the replacement's effect on what the layer passes on; absorption can be partial, so intermediate quantities may still change. Appendix Table C.1 collects the terms.

3.1 Four layers and their decision rules

Execution asks whether the replacement reaches the computation. The *supplied graph* is passed to the model; the *executed edges* are the message edges its layers actually read, which differ from the supplied graph when a detector learns its own, *native*, graph. We freeze the checkpoint and record the executed edges and the predictions before and after the replacement. If neither changes, the ablation tested nothing, and we move the intervention to the executed edges. *Information* asks whether the replaced input carries predictive information beyond self-history, and whether the replacement keeps it. We estimate on normal data the relative gain g of the original graph's history over self-history, and the gain g_{rep} that the replacement still carries. *Utilization* asks whether the fitted model uses the available information. We compare the detector's normal risk with its graph against the same detector restricted to self-loops, $e = 1 - \mathrm{MSE}_{\mathrm{graph}}/\mathrm{MSE}_{\mathrm{self}}$, and we inspect which sources it selects and how it shares attention. *Scoring* asks whether a change in prediction reaches the anomaly ranking. We trace it through residuals, score aggregation, and AP relative to prevalence. A replacement *reaches the ranking* when its AP contrast is resolved (the interval over replicate units excludes zero; for one run, $|\Delta\mathrm{AP}| > 0.005$), and it takes a *scoring route* when AP changes without a measured gain in mean normal risk; predictions may still change, because equal mean risk can hide different residuals.

Algorithm 1 turns these measurements into labels with one tolerance τ (1% throughout). The layers are read in order and the first absorbing layer gives the label, with one exception: when an information- or utilization-limited replacement still moves AP, the label becomes a scoring route, whereas retention already allows a partial AP change. If neither layer absorbs it ($g \geq \tau$, $e \geq \tau$), the normal-data label is *graph used*. The labels are operational for the stated probes, budget, and tolerance rather than a causal decomposition: *information retained* means a measurable gain in the replacement history, not the original information, and an unresolved AP contrast leaves open whether ranks changed. Each label points to a replaceable component; Section 4 tests the rules, and Appendix D.2 turns them into a five-check report.

Algorithm 1 Diagnose a graph intervention

Require: Detector; normal and test data; interventions, probes, score; risk tolerance τ (1%); AP tie band 0.005

1. **Execution:** with checkpoint and calibration frozen, record executed edges and predictions before and after the replacement. *Unchanged: the replacement never reached the model; intervene on the executed edges.*
2. **Information:** on normal data, estimate the relative gain g of supplied history over self-history (Equation (1)) for the original and the replaced graph. $g < \tau$*: information-limited. Replaced graph keeps* $g \geq \tau$*: information retained; locate the carrying lags or sources by removal.*
3. **Utilization:** compare normal risk with the graph against self-only support; mask critical and control lags on executed source edges, frozen and refitted; record selected sources and attention shares. $g \geq \tau$ *but the graph lowers normal risk by less than* τ*: utilization-limited; replace the located component and repeat.*
4. **Scoring:** trace prediction changes into residuals, scores, ranks, AP with prevalence, AUROC, and alarms. *Risk changes but AP does not: absorbed at scoring. AP changes without a measured normal-risk gain: a scoring route, which replaces an information- or utilization-limited label.*

Ensure: The first absorbing layer, every layer's verdict, the located component, and the effect of its replacement.

3.2 MEASURING THE INFORMATION LAYER

The information layer compares two predictive risks. For a zero-mean stationary Gaussian process, partition inputs into self-history H and supplied history Z, with normal covariance C_{UV}. Let $K = C_{ZZ} - C_{ZH}C_{HH}^{\dagger}C_{HZ}$ and $r_c = C_{ZY} - C_{ZH}C_{HH}^{\dagger}C_{HY}$. The conditional-projection identity gives

$$\Delta_{\text{pred}} = R^*_{\text{self}} - R^*_{H,Z} = r_c^{\top} K^{\dagger} r_c, \qquad \Delta_{\text{pred}} = 0 \iff r_c = 0. \tag{1}$$

On data, we estimate the relative gain $g = \Delta_{\text{pred}}/R^*_{\text{self}}$ with per-node ridge predictors fitted on normal data before detector training, checked with blocked cross-validated ridge and gradient boosting. A replacement keeps a positive gain in two ways. Older history of the new neighbors can carry what the removed edges carried (*retention*), and many sources can share it (*redundancy*).

3.3 A TESTBED WITH EXACT PREDICTIVE REFERENCES

To measure each layer against a known answer, we construct paired regimes that change cross-channel relations while keeping every channel's own process. Adjacency rows denote receivers and columns senders. With $y_t \in \{0, 1\}$ the regime and P_0, P_1 parent permutation matrices,

$$x_t = \rho P_{y_t} x_{t-1} + \sqrt{1-\rho^2}\epsilon_t, \qquad \epsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0, I_d), \quad 0 < \rho < 1, \tag{2}$$

started from $x_0 \sim \mathcal{N}(0, I_d)$ independently of the regimes. Then $x_t \sim \mathcal{N}(0, I_d)$, and the fixed-regime lag covariance is $\Gamma_k = \rho^k P^k$. A detector reads its self-history and one aggregation of its supplied neighbors over a window of W steps, and we score the node-averaged squared residual.

Proposition 1 (Fixed-channel equivalence). If each node has the same cycle length under P_0 and P_1, its finite-dimensional distributions agree in the two stationary regimes, while lag-one cross-channel covariances differ whenever $P_0 \neq P_1$.

Proposition 2 (First predictive lag of a reversed neighbor). In a normal m-cycle, $m \geq 3$, the reversed graph supplies target i with its normal child j. Let $L = m - 1$. For $W < L$ the normal conditional mean given the interface is zero; at $W = L$ it is $\rho^L x_{j,t-L}$, with normal MSE $1 - \rho^{2L}$ and MSE $1 + \rho^{2L}$ in the reversed regime.

Both follow from Γ_k (Appendix A). A score on one fixed channel's window has the same distribution in both regimes, so the specified detector separates them only through cross-channel history (Appendix A.1), and the information layer should find reversed-graph gain first at lag $m - 1$. Because the residuals are Gaussian, we also compute the exact AP of the population MSE-optimal forecaster for each graph (Appendix B.1), a reference rather than the best AP any detector could reach.

Interventions and fitting protocols. A *refit* trains and evaluates on each supplied graph; a *frozen* replacement changes the test graph alone. Random graphs match degree, and contrasts are paired within data seeds. To test use of an exposed lag, we mask on the executed source edge either the *critical* lag, at which the source predicts the target (1 for a parent, $m - 1$ for a child), or a *control* lag of the same source. We report AP without point adjustment, with prevalence.

4 VALIDATING THE DIAGNOSIS ON PLANTED ABSORPTION

Before we apply the diagnosis where the absorber is unknown, we test whether its rules name the right layer when the answer is known. We build twelve graph replacements, each with an absorber planted by construction (Appendix Table D.1). Three plants act at execution (detectors that ignore or symmetrize the supplied graph), two remove the information and two retain it (history retention; common factors), two limit utilization (GDN's top-k selector; static attention among distractors), one acts at scoring (noise bursts that the score ranks near-perfectly with or without the graph), and two controls contain none. Every case uses complete GDN or MTGNN, ten data seeds, and prevalence 0.10. The rules are those of Algorithm 1 as released in our toolkit, committed before the benchmark was designed. We committed the protocol and the truth key before any fit, locked all normal-data measurements before opening any test, and ran the rules without the key. We designed the plants after the rules were fixed, each with a single absorber, so the benchmark tests whether the rules name a known absorber, not how often each absorber occurs in practice.

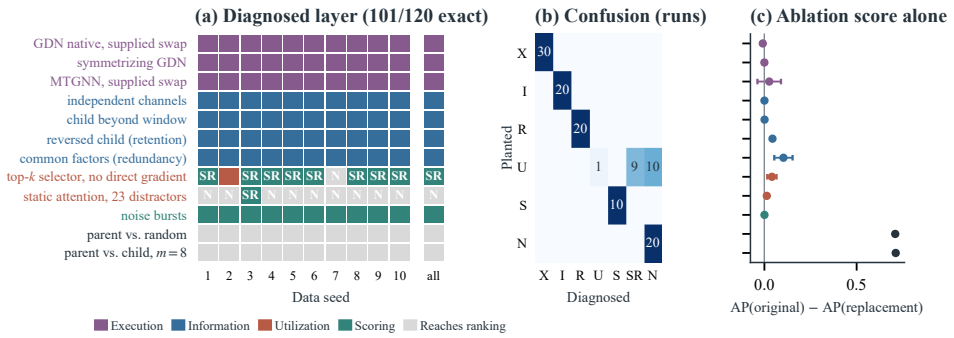


Figure 2: **The diagnosis names planted absorbers that the ablation score cannot tell apart.** (a) Diagnosed layer for each planted case (rows, colored by planted layer) and data seed; letters mark labels that differ from the planted one (SR: scoring route, N: reaches ranking). (b) Planted against diagnosed label over all runs. (c) The refitted ablation contrast alone (mean and 95% interval over seeds); shaded: ± 0.005.

What the rules distinguish. Read alone, the ablation contrast stays within ± 0.005 for four cases whose absorbers lie at three different layers: execution (P02), information (P04, P05), and scoring (P10). Retained information absorbs part of a replacement: AP falls by 0.04 and 0.10, against 0.71 for the controls (Figure 2c). Our fixed rules separate these cases. In every seed they recover every execution, information, retention, and scoring plant, and both controls reach the ranking, ten of the twelve cases (Figure 2a,b). Readouts of graph importance confound the cases. For the symmetrized, retained, and saturated graphs, removing all non-self messages raises normal MSE by 52.5–89.4% and attention puts at least 94% of its weight on graph sources, while the refitted ablation moves AP by at most 0.044 (Appendix Table D.3). Even trained with the planted labels, a classifier on occlusion, attention, permutation importance, conditional randomization importance, or GraphMask names the absorbing layer of an unseen case in at most 51% of runs, against 84% for the blind rules (Appendix D.3). The labels are identical for three information estimators and every $\tau \in \{0.5, 1, 2, 5\}\%$; counting an AP change only when its interval excludes zero lowers accuracy to 76%.

Where the rules fail: partial utilization. Over all runs, the rules recover the planted label in 101 of 120 runs. All nineteen misses fall in the two utilization plants, which yield one exact match in twenty runs. In these two cases the planted component limits the use of the graph while leaving the graph executed. GDN's top-k selection still exposes a few parents: the utilization layer flags its graph in 9 of 10 seeds, but the replacement still moves AP (median 0.03), so the final label becomes a scoring route. Static attention still extracts a 1.3% normal-risk gain from the parent among 23 distractors, just above τ. A single label can thus hide a partial utilization limit; we keep every layer's verdict and confirm such labels by replacing the located component (Sections 5.2 and 6).

5 Case Study: Locating Absorption in a Controlled Cycle

We apply the diagnosis on the testbed of Section 3.3: directed four-cycles ($d = 64$, $\rho = 0.6$) with reversed regimes as anomalies. Graph-GRU (Cho et al., 2014) and Graph-TCN (Bai et al., 2018) probe the information layer; GDN and MTGNN (Wu et al., 2020) probe execution, lag use, and selection. Models train on normal data, and intervals are over data seeds.

5.1 Execution and Information: Discarded Graphs and Retained History

In official GDN, embedding top-k selection determines the message edges. When we swap the supplied graph, the executed edges, the predictions, and AP (0.537) stay unchanged, so the swap is absorbed at execution. FuSAGNet (Han & Woo, 2022), MTGNN, and GraGOD's GDN behave the same way when they learn the graph, GANF (Dai & Chen, 2022) uses the supplied graph to initialize a learned adjacency, and GTA (Chen et al., 2022) and GReLeN (Zhang et al., 2022) infer their graphs internally (Appendix Table E.1). Rerunning GraGOD's released SWaT pipeline confirms

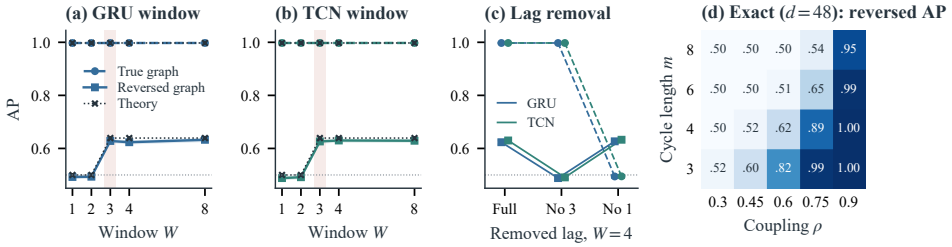


Figure 3: **History retention predicts when and how much a reversed graph still detects.** (a,b) Graph-GRU and Graph-TCN AP versus window W for true (dashed) and reversed (solid) graphs; crosses: population MSE-optimal forecaster's AP. (c) Refit at $W = 4$ without one neighbor lag. (d) Exact AP of that forecaster, reversed graph, $W = 8$ ($d = 48$). Ten data seeds in (a–c); 95% intervals.

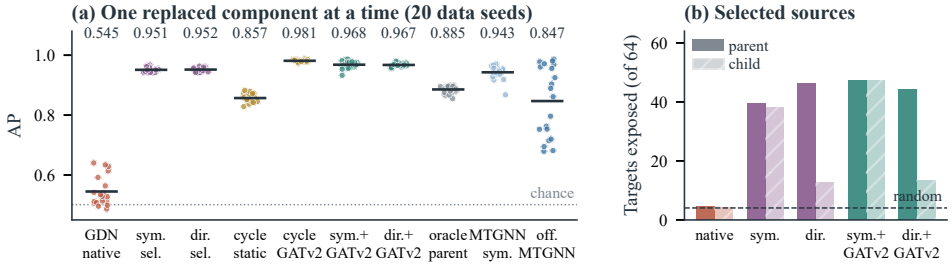


Figure 4: **Replacing one GDN component at a time on the directed four-cycle.** (a) AP over 20 data seeds for native GDN, trainable selectors, the cycle as fixed sources under static or GATv2 attention, both replacements, an oracle parent, and MTGNN with symmetric or official graph constructor. (b) Targets whose sources include the parent (solid) or child (hatched); dashed: random.

this (a swap changes predictions by at most 2.9×10^{-6}) and exposes a second execution absorber: its GDN and GCN reshape rather than transpose their input, so each node receives a mix of sensors' values in place of its own history. In 17 of 18 configurations, AP stays within 0.717–0.775 as normal MSE ranges from 0.15 to 1.32. Every topology-minus-random contrast of GDN stays unresolved (Appendix E.2). When we instead reverse the executed edges of a parent-trained GDN, the reversal reaches the ranking and inverts it (AP 0.929 to 0.343).

Reversed-neighbor history becomes useful exactly at the predicted lag. Proposition 2 predicts that reversed-neighbor history first contributes at lag 3. As we widen the window from 2 to 3, Graph-GRU's reversed-graph AP jumps from chance to 0.628, while true-graph AP stays near its maximum (Figure 3a,b). We then refit at $W = 4$ after masking a single neighbor lag. Removing lag 3 returns reversed-graph AP to chance (0.488), removing lag 1 leaves it unchanged, and the true graph shows the mirror image (Figure 3c). Trained GRU is within 0.016 of the MSE-optimal forecaster's AP at every window, jump included. In the exact map (Figure 3d), across cycle lengths 3–8 and couplings 0.3–0.9, reversed-graph AP first exceeds chance at $W = m - 1$ and grows with ρ^{m-1}; at $\rho = 0.9$ and $m \leq 4$ it reaches 1.00. The first predictive lag persists with heavy tails and tanh dynamics (Appendices B.2–B.3).

5.2 Utilization: where GDN loses the cycle

We first give GDN one oracle source, taken from the generator, and find that GDN uses the critical lag: masking the parent's lag 1 drops AP from 0.973 to chance, masking lag 3 leaves it unchanged, and a child source shows the mirror image (Appendix Table E.4). With its own graph, however, GDN detects near chance, and we find the learned graph utilization-limited: its normal MSE (0.986) matches that of the population-optimal self-history predictor ($1 - \rho^8 = 0.983$). The selection explains why. About four of 64 targets see their parent, the rate of four sources drawn at random (4.25 against 4.07), because GDN chooses sources by the top-k cosine similarity of node embeddings. The embeddings are detached before this ranking, so the choice lacks a direct gradient. They still train

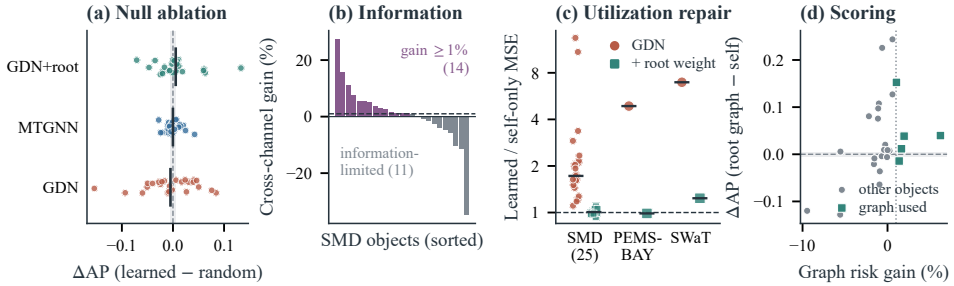


Figure 5: **Repairing normal prediction while the graph-ablation AP contrast stays unresolved.** Points are SMD objects (means over 3 initializations). (a) Learned minus random AP. (b) All-channel ridge gain over self-history (dashed: 1%). (c) Learned-graph over self-only normal MSE, with and without the root-weight repair, also for PEMS-BAY and SWaT. (d) After the root-weight repair: risk gain vs. AP gain; squares: graph used. The three-component replacement is reported in the text.

through attention and the output layer, yet parents stay exposed at the random rate. We therefore replace one component at a time and keep data, budget, and score fixed (Figure 4).

Selection. We replace the top-k choice alone by a selector whose scores enter the attention logit, which gives these scores a direct loss gradient (the top-k indices stay discrete) and also changes the attention parameterization. Parents are then exposed to about 40 of 64 targets, and AP rises from 0.545 to 0.951 on every seed. Direction in the selector score is unnecessary. A symmetric score exposes children as often as parents, yet a directed score gives the same AP. Making MTGNN's antisymmetric graph constructor symmetric changes its AP from 0.847 to 0.943.

Attention. We next give GDN the cycle as fixed sources, and its static attention reaches AP 0.857. Static attention scores a source by a term that is independent of the target (Brody et al., 2022). All targets of a cycle therefore rank their candidates identically, and at most one target per cycle can give its parent the largest weight (Proposition 4, Appendix E.4). With GATv2 attention scores alone, the parent receives 0.95 of the weight, normal MSE reaches its population optimum, and AP rises to 0.981 on every seed. With a trainable selector as well, AP is 0.967. Both repairs hold at prevalence 0.05 and under a second initialization on every seed. Native GDN thus detects near chance primarily because its source selection lacks a direct gradient, and secondarily because its attention is static.

6 Case Study: Diagnosing a Null Ablation on SMD

When a learned-versus-random AP contrast is unresolved on real data, can the diagnosis still find and repair a defect, and does the repair change the original contrast? We use the 25 SMD objects (Su et al., 2019) held out from development, with 38 channels each. For each object we refit GDN with three initializations on its learned, self-only, and degree-matched random graphs, each also with longer training. We fix every graph, checkpoint, calibration, and normal-data verdict before any test score; intervals resample objects.

The endpoint is a null ablation. The learned graph's AP is 0.005 below the random graph's, with an interval spanning zero, and official MTGNN agrees (Figure 5a). In all 75 learned checkpoints, swapping the supplied graph leaves every executed edge unchanged, so we intervene on executed edges below. On normal data, the history of all 38 channels (in population, an upper bound on any supplied subset) adds less than 1% to self-history on 11 objects, labeled information-limited by the primary probe; on the other 14 the gain is larger (Figure 5b). Probes on each checkpoint's executed sources give median gains below 4% (Appendix F.2). Across tolerances and four further estimators, the information-limited count ranges from 5 to 20, and GDN reaches graph use on none (Appendix F.3).

The null endpoint hides a prediction deficit. On the 14 objects with cross-channel gain, GDN leaves this gain unrealized, and on all 25 objects its learned graph predicts normal data worse than self-loops alone (normal MSE 0.368 against 0.194; Figure 5c). Better graphs or more capacity leave

this gap in place. GDN's single softmax gives a node's own history a share of 0.21, as if uniform over its five inputs, though this share is unrelated to the excess error (pre-registered; Spearman 0.04).

Replacing the located component repairs prediction. We carry self-history through a separate root weight outside the softmax and change nothing else. The repair closes at least 91% of the gap on every SMD object and brings normal MSE to the self-only level. The graph is then used on five objects, and the rank correlation of cross-channel gain with normal-risk gain rises from 0.35 to 0.48. With the trainable selector and GATv2 attention of Section 5.2 added to the root weight, the graph is used on 9 objects, again at self-only normal MSE.

The original contrast remains a separate question. Three AP comparisons answer three questions. Does the repair improve detection over the original model? Repaired AP (0.380) stays close to the original (0.394), a paired change whose interval spans zero, and the three-component replacement gives -0.036, also spanning zero. Does the learned topology beat random support? The contrast stays unresolved after the root weight ($+0.005$) and after all three replacements (-0.014; Appendix F.3), and random sources lower normal risk like learned ones, consistent with redundancy, the cross-channel counterpart of history retention. Does cross-channel support beat self-history? The repaired learned graph gives $+0.034\,[+0.0004, +0.069]$ over self-only GDN, close to the unrepaired $+0.048\,[-0.005, +0.116]$. Per object (Figure 5d), AP rises on four of the five graph-using objects of the root-weight repair, but as much where normal risk is unchanged. The null endpoint thus hid a graph that degraded prediction, which a located component repaired; whether the learned topology matters for detection is a separate question.

Other datasets. On SWaT (Goh et al., 2017), MTGNN's learned-graph advantage over self-loops (0.25 AP in three initializations) shrinks to 0.095 with ten initializations, interval spanning zero, and to 0.004 under a channel-mean score, so it depends on aggregation and initialization (Appendix F.4); GDN stays utilization-limited despite a root weight. On PEMS-BAY traffic sensors (Li et al., 2018), a root weight lowers the learned-graph to self-only error ratio from 4.9 to 0.99, every detector ranks injected replays near prevalence, and graph use is absorbed at scoring.

7 Discussion

What the diagnosis adds to a null result. An unexecuted graph, node inputs misaligned with the graph, a graph that degrades normal prediction, and a graph whose information other channels also carry all leave the endpoint unchanged and would all be read as an irrelevant graph. On SWaT and SMD the diagnosis replaced this single reading with located, testable statements rather than a higher AP, each naming an existing component to replace: a loss-trained selector, dynamic attention (Brody et al., 2022), or a root weight for self-history as in GraphSAGE (Hamilton et al., 2017).

Scope and limitations. Propositions 1 and 2 hold for the permutation VAR; multiple paths change the critical lag in general graphs (Records R9.1 and R9.2), and in random sparse VARs a reversed graph retains little (AP at most 0.53 at $W = 4$; Appendix B.3), so cycles are a testbed with exact references rather than a model of sensor graphs. Labels depend on the probes, budget, and tolerances, and the planted cases test whether the rules name a known single absorber (Section 4). Per-window detectors such as GTA and GReLeN need the execution check per window. The SMD test labels were opened earlier in an adoption study (Record R9.5), and SWaT has one test series at one-minute resolution (Appendices F.1, F.4, G.1).

8 Conclusion

Similar graph-ablation outcomes can arise from absorption at execution, information, utilization, or scoring; layered graph-ablation diagnosis distinguishes them and names the component to examine next, and history information retention keeps a reversed neighbor predictive from lag $m - 1$. Blind rules recovered 101 of 120 planted labels; located replacements raised GDN's cycle AP from 0.545 to 0.967 and repaired an SMD prediction deficit hidden by a null ablation, with no resolved AP change over the original model.

REFERENCES

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.nips.cc/paper/2018/hash/294a8ed24b1ad22ec2e7efea049b8737-Abstract.html.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv preprint arXiv:1803.01271*, 2018. URL https://arxiv.org/abs/1803.01271.

Federico Bello, Gonzalo Chiarlone, Marcelo Fiori, Gastón García González, and Federico Larroca. GNNs for time series anomaly detection: An open-source framework and a critical evaluation. In *Proceedings of the 15th International Conference on Pattern Recognition Applications and Methods*, pp. 53–61. SCITEPRESS, 2026. doi: 10.5220/0014312000004067. URL https://www.scitepress.org/PublishedPapers/2026/143120/.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

Zekai Chen, Dingshuo Chen, Xiao Zhang, Zixuan Yuan, and Xiuzhen Cheng. Learning graph structures with Transformer for multivariate time-series anomaly detection in IoT. *IEEE Internet of Things Journal*, 9(12):9179–9189, 2022. doi: 10.1109/JIOT.2021.3100509.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

Enyan Dai and Jie Chen. Graph-augmented normalizing flows for anomaly detection of multiple time series. In *International Conference on Learning Representations*, 2022.

Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4027–4035, 2021. doi: 10.1609/aaai.v35i5.16523. URL https://ojs.aaai.org/index.php/AAAI/article/view/16523.

Francis X. Diebold and Roberto S. Mariano. Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):253–263, 1995. doi: 10.1080/07350015.1995.10524599.

Astha Garg, Wenyu Zhang, Jules Samaran, Ramasamy Savitha, and Chuan-Sheng Foo. An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2508–2517, 2022. doi: 10.1109/TNNLS.2021.3105827.

Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. A dataset to support research in the design of secure water treatment systems. In *Critical Information Infrastructures Security (CRITIS 2016)*, volume 10242 of *Lecture Notes in Computer Science*, pp. 88–99. Springer, 2017. doi: 10.1007/978-3-319-71368-7_8.

C. W. J. Granger. Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica*, 37(3):424–438, 1969. doi: 10.2307/1912791. URL https://doi.org/10.2307/1912791.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, pp. 1024–1034, 2017.

Lu Han, Han-Jia Ye, and De-Chuan Zhan. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):7129–7142, 2024. doi: 10.1109/TKDE.2024.3400008.

Siho Han and Simon S. Woo. Learning sparse latent graph representations for anomaly detection in multivariate time series. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2977–2986, 2022. doi: 10.1145/3534678.3539117.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Siwon Kim, Kukjin Choi, Hyun-Soo Choi, Byunghan Lee, and Sungroh Yoon. Towards a Rigorous Evaluation of Time-Series Anomaly Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7194–7201, 2022. doi: 10.1609/aaai.v36i7.20680. URL https://doi.org/10.1609/aaai.v36i7.20680.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113 (523):1094–1111, 2018. doi: 10.1080/01621459.2017.1307116.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJiHXGWAZ.

Zhixun Li, Liang Wang, Xin Sun, Yifan Luo, Yanqiao Zhu, Dingshuo Chen, Yingtao Luo, Xiangxin Zhou, Qiang Liu, Shu Wu, Liang Wang, and Jeffrey Xu Yu. GSLB: The graph structure learning benchmark. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 36, pp. 30306–30318, 2023. doi: 10.52202/075280-1319.

Pablo Morales-Mombiela, Daniel Hernández-Lobato, and Alberto Suárez. Statistical tests for the detection of the arrow of time in vector autoregressive models. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pp. 1544–1550, 2013. URL https://www.ijcai.org/Proceedings/13/Papers/230.pdf.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2211.14730.

Marc Pinet, Julien Cumin, Samuel Berlemont, and Dominique Vaufreydaz. Anomalies in multivariate time series benchmarks are mostly univariate. In *12th Mining and Learning from Time Series Workshop (MiLeTS at KDD 2026)*, 2026. URL https://arxiv.org/abs/2606.02670. Workshop paper.

Bastian Prasse and Piet Van Mieghem. Predicting network dynamics without requiring the knowledge of the interaction graph. *Proceedings of the National Academy of Sciences*, 119(44): e2205517119, 2022. doi: 10.1073/pnas.2205517119.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for NLP with differentiable edge masking. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=WznmQa42ZAx.

Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly Detection in Time Series: A Comprehensive Evaluation. *Proceedings of the VLDB Endowment*, 15(9):1779–1797, 2022. doi: 10.14778/3538598.3538602. URL https://doi.org/10.14778/3538598.3538602.

Harshay Shah, Prateek Jain, and Praneeth Netrapalli. Do input gradients highlight discriminative features? In *Advances in Neural Information Processing Systems*, volume 34, pp. 2046–2059, 2021.

Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9:307, 2008. doi: 10.1186/1471-2105-9-307.

Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2828–2837, 2019. doi: 10.1145/3292500.3330672. URL https://doi.org/10.1145/3292500.3330672.

Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily B Fox. Neural Granger Causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4267–4279, 2022. doi: 10.1109/tpami.2021.3065601. URL https://doi.org/10.1109/tpami.2021.3065601.

Brian D. Williamson, Peter B. Gilbert, Noah R. Simon, and Marco Carone. A general framework for inference on algorithm-agnostic variable importance. *Journal of the American Statistical Association*, 118(543):1645–1658, 2023. doi: 10.1080/01621459.2021.2003200.

Irene Winkler, Danny Panknin, Daniel Bartz, Klaus-Robert Müller, and Stefan Haufe. Validity of time reversal for testing Granger causality. *IEEE Transactions on Signal Processing*, 64 (11):2746–2760, 2016. doi: 10.1109/TSP.2016.2531628. URL https://arxiv.org/abs/1509.07636.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 753–763, 2020. doi: 10.1145/3394486.3403118. URL https://doi.org/10.1145/3394486.3403118.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1eBeyHFDH.

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*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/d80b7040b773199015de6d3b4293c8ff-Abstract.html.

Weiqi Zhang, Chen Zhang, and Fugee Tsung. GReLeN: Multivariate time series anomaly detection from the perspective of graph relational learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 2390–2397, 2022. doi: 10.24963/ijcai.2022/332. URL https://www.ijcai.org/proceedings/2022/332.

Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Multivariate time-series anomaly detection via graph attention network. In *IEEE International Conference on Data Mining (ICDM)*, pp. 841–850, 2020. doi: 10.1109/ICDM50108.2020.00093.

Zhiyao Zhou, Sheng Zhou, Bochao Mao, Xuanyi Zhou, Jiawei Chen, Qiaoyu Tan, Daochen Zha, Yan Feng, Chun Chen, and Can Wang. OpenGSL: A comprehensive benchmark for graph structure learning. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 36, pp. 17904–17928, 2023. doi: 10.52202/075280-0787.

Reproducibility Statement

Anonymized code and the supplementary records are provided as supplementary material with this submission. The code contains a lightweight, detector-agnostic implementation of the layered diagnosis (the four layer checks and the decision rules of Algorithm 1, with tests and an example that reproduces the paper's layer labels from saved results), together with the data generators, detector adapters and intervention runners, analysis and plotting scripts, and the protocols of the studies whose design was committed before their formal runs; scripts regenerate every reported metric from saved predictions. The appendix gives the proofs, generator settings, intervention grids, training budgets, pinned versions of third-party detectors, and scoring and calibration protocols, and the supplementary records retain per-seed and per-object tables, development records, corrections, and original failed attempts (Appendix H lists their contents).

Complete GDN experiments retain executed edges, calibrations, and both score streams; selective-lag studies retain the fixed-source and native fits and the frozen-mask predictions; degree-matched controls retain checkpoints, graphs, and per-seed metrics for every arm. For the SMD, SWaT, and

PEMS-BAY studies, every graph, checkpoint, calibration, and layer label was fixed from normal data before test scores were computed, and the SMD test labels had already been opened in a preliminary study of predictor adoption (Record R9.5). The SWaT data are available from their owners on request and are not redistributed; the PEMS-BAY replay attacks are regenerated from a fixed seed.

USE OF AI

Generative AI assisted method checks, algebraic calculations, experiment and plotting code, and language and typesetting revisions. Derivations and analyses were checked through numerical verification and code tests. The authors are responsible for the accuracy, provenance, and interpretation of the final submission.

APPENDIX
WHEN GRAPH ABLATIONS CHANGE NOTHING: LOCATING WHERE TIME-SERIES ANOMALY DETECTORS ABSORB A GRAPH REPLACEMENT

Anonymous authors
Paper under double-blind review

The appendix follows the order of the main text: proofs (A), mechanism experiments (B), setup and diagnostic rules (C), the planted-absorption benchmark (D), complete detectors (E), real-data studies (F), extended related work and discussion (G), and reproducibility (H). Items numbered with an R are in the supplementary records described in Appendix H.

A HISTORY AND PREDICTIVE RISK: SUPPLEMENTARY PROOFS

This section proves Propositions 1 and 2 and the conditional prediction results, and explains how predictive gains affect squared-residual distributions. We first establish fixed-channel process equivalence and return lags on cycles, then derive conditional projections for general visible histories. We finally distinguish prediction objectives under identical history, frozen graph replacement, and bidirectional support.

A.1 FIXED-CHANNEL PROCESS EQUIVALENCE

Throughout these proofs, the regime sequence is prespecified or independent of the initial state and innovations. Finite-dimensional distributions are compared only over windows lying entirely within one regime.

Proof of Proposition 1. A permutation matrix satisfies $PP^\top = I_d$. By Equation (2) and independent Gaussian innovations, if $x_{t-1} \sim \mathcal{N}(0, I_d)$, then

$$\mathrm{Cov}(x_t) = \rho^2 PP^\top + (1-\rho^2) I_d = I_d. \tag{A.1}$$

Stationary initialization gives zero means and covariance I_d at every time. Iterating k steps yields

$$\Gamma_k = \mathrm{Cov}(x_t, x_{t-k}) = \rho^k P^k, \qquad k \geq 0. \tag{A.2}$$

Let $c_P(i)$ denote the cycle length of node i under P. The permutation returns to i after k steps exactly when $c_P(i)$ divides k, so

$$\mathrm{Cov}(x_{i,t}, x_{i,t-k}) = \rho^{|k|} \mathbf{1}\{c_P(i) \mid |k|\}. \tag{A.3}$$

If $c_{P_0}(i) = c_{P_1}(i)$, this channel has the same mean and covariance at every finite collection of times under both regimes. Joint Gaussianity implies equal finite-dimensional distributions. Since $\rho > 0$ and $P_0 \neq P_1$, we have $\rho P_0 \neq \rho P_1$, so lag-one cross-channel covariances differ. This proves the proposition.

General conjugation $P_1 = QP_0Q^{-1}$ preserves cycle type but need not preserve each labeled node's cycle length. For example, take $P_0 = (1\,2)(3\,4\,5)$ and $Q = (2\,3)$; then

$$P_1 = QP_0Q^{-1} = (1\,3)(2\,4\,5). \tag{A.4}$$

Node 2 moves from a two-cycle to a three-cycle, changing its lag-2 autocovariance from ρ^2 to 0. If all nodes lie in cycles of length m, any conjugation preserves each node's cycle length and satisfies the proposition.

For a fixed scoring function reading only a stationary window of one fixed channel, the two population score distributions agree. Population AUROC is therefore 0.5, and population AP equals

anomaly prevalence. If the score is fitted on independent normal training data, the result applies conditionally on that data. Equal univariate distributions do not imply an equal joint distribution of multiple channel scores: cross-channel dependence may still change their aggregate. We therefore measure channel-independent AR with aggregated residuals separately as an empirical baseline.

Boundary guards. Even with switching P_{y_t}, the marginal distribution remains $\mathcal{N}(0, I_d)$ at each time. Covariances spanning switches contain mixed permutation products, so Equation (A.3) no longer applies directly. Let $b \geq W$ be the initial test exclusion and $\mathcal{T}$ the regime-switch target times. Metrics use

$$\mathcal{I}_{g,b} = \{b \leq t < T_{\text{test}} : \ t \notin [\tau, \tau + g) \text{ for all } \tau \in \mathcal{T}\}, \qquad g \geq W. \tag{A.5}$$

Linear controls and linear lag interventions use $g = 4$ or common $g = 8$, specified by experiment. Main neural experiments use $g = b = 8$; supplementary per-window grids use $g = b = W$. These sets are reported separately rather than aligned pointwise. Each window enters the network independently; GRU hidden states do not persist across windows. Switch times come from generated labels to exclude mixed-regime histories. This evaluation does not measure alarm delay when switch locations are unknown.

Connection to time reversal. Direction reversal here coincides exactly with classical time reversal. A stationary Gaussian VAR has reverse coefficient $CB^\top C^{-1}$. Substituting $C = I, B = \rho P$ gives $\rho P^\top$, with reverse innovation covariance $(1 - \rho^2)I$ (Morales-Mombiela et al., 2013; Winkler et al., 2016). Thus the fixed reversed regime is the time reversal of the normal stationary process. General matched rewiring lacks this correspondence, and a switching sequence is not simply normal data played backward. This time-reversal construction lets us characterize when the history available to a model first becomes predictive.

B Mechanism Experiments: History, Targets, and Aggregation

This section tests whether models use critical history, whether identical inputs support different targets, and how local changes affect system-level scores. Training settings precede the corresponding ablations and multi-parent results.

B.1 Score distributions under local reversal

Conditional-gain counterexamples in general graphs and two-parent cycles are retained in Record R9; this subsection derives the score laws behind the population references of the main text.

Let d be divisible by four, with $d/4$ independent normal four-cycles. Here $k \in \{1, \ldots, d/4\}$ counts **fully reversed four-cycles**, affecting $4k$ channels. It is neither an edge-change count nor the count used for general within-block matched rewiring. Fix either the true-graph one-step population predictor or the reversed-graph three-step predictor, writing $r = \rho^2$ or $r = \rho^6$, respectively.

Unreversed cycles have residual covariance $(1-r)I_4$. For a reversed cycle with normal permutation P, Equation (R13.12) gives $(1+r)I_4 - 2rP^2$. Since P^2 has eigenvalues $+1, -1$ twice each, residual eigenvalues are $1-r$ twice and $1+3r$ twice. Independent innovations and initial states across blocks yield the exact score law

$$S_0 \overset{\text{d}}{=} \frac{1-r}{d}\chi^2_d, \qquad S_1 \overset{\text{d}}{=} \frac{(1-r)\chi^2_{d-2k} + (1+3r)\chi^2_{2k}}{d}, \tag{B.1}$$

$$\mathbb{E}[S_1] - \mathbb{E}[S_0] = \frac{8kr}{d}, \qquad \text{Var}(S_0) = \frac{2(1-r)^2}{d}, \tag{B.2}$$

$$\text{Var}(S_1) = \frac{2\{(d-2k)(1-r)^2 + 2k(1+3r)^2\}}{d^2}. \tag{B.3}$$

The two anomalous chi-square variables in Equation (B.1) are independent. Mean score change divided by normal standard deviation is $8kr/[\sqrt{2d}(1-r)]$: it grows as $\sqrt{d}$ under global reversal $k = d/4$, but decays as $d^{-1/2}$ for fixed k. Increasing dimension can improve detection of widespread

changes while diluting local ones. AP also depends on the full distribution and anomaly prevalence, not just this ratio.

At $d = 64, \rho = 0.6$, independent spectral sampling gives reversed-graph AP 0.6389 for global reversal and 0.5082 for one cycle. Population predictors on ten independent switching trajectories give 0.6372 and 0.5034. Each spectral condition uses 20,000 samples per class and ten Monte Carlo repetitions. GRU reuses the same normal checkpoint across reversal extents (Table R8.3), with ten data seeds per condition.

Exact population references for the main text. For global reversal at prevalence $1/2$, population AP is $\int f_1(s)\,\bar{F}_1(s)/\{\bar{F}_1(s) + \bar{F}_0(s)\}\,ds$, where f_1 and $\bar{F}_y$ are the density and survival functions of the score laws in Equation (B.1) with $k = d/4$; this is the large-sample limit of average precision. Numerical integration at $d = 64$, $\rho = 0.6$ gives 0.6391 for the reversed-graph three-step predictor ($r = \rho^6$) and 0.9975 for the true-graph one-step predictor ($r = \rho^2$). These are the population-optimal predictors at $W = 3$ (Proposition 2) and $W = 4$ (Record R13.2); at $W = 8$ the child's lag-3 state is still the nearest visible state on the target's parent chain, so the remaining innovations are independent of all visible history (the argument of Appendix E) and the predictor is unchanged. For the reversed graph at $W \leq 2$ the optimal predictor is zero, the two score laws coincide, and AP is exactly 0.5. These values are the crosses in Figure 3(a,b). Two checks agree: 4×10^6 draws per class from the same laws give 0.6389–0.6396 and 0.9975, and shared least-squares predictors fitted on simulated $d = 64$ trajectories (20,000 normal training steps, 200,000 test steps per regime) give a mean reversed-graph AP of 0.640 over 13 simulation seeds (range 0.634–0.645) at $W = 3$ and the same values at $W = 4$. At $d = 48$ the integration gives 0.6203 and 0.9926. Trained Graph-GRU reaches 0.4935, 0.6281, and 0.6234 at $W = 2, 3, 4$ (seed intervals $[0.4852, 0.4997]$, $[0.6202, 0.6352]$, $[0.6151, 0.6304]$), 0.007–0.016 below the reversed-graph references, and 0.9973–0.9974 on the true graph, whose intervals contain 0.9975; Graph-TCN lies 0.008–0.013 below the reversed-graph references.

B.2 Population retention map ($d = 48$)

This analysis asks how much detection a reversed graph retains before any model is trained. For each cycle length $m \in \{3, 4, 6, 8\}$, coupling $\rho \in \{0.3, 0.45, 0.6, 0.75, 0.9\}$, and window $W = 1, \ldots, 8$, we take Equation (2) with $d = 48$ channels, the unidirectional interface (own lags and supplied-neighbor lags $1, \ldots, W$), the population-optimal predictor fitted on normal data and shared across nodes, and the node-averaged squared residual as score; AP compares the reversed with the normal regime at prevalence $1/2$, each regime stationary. The map is computed in closed form. Cycles are independent and exchangeable, so the residual vector of one cycle is a linear map of jointly Gaussian lags with covariance $\Gamma_k = \rho^k P^k$; the score is therefore a generalized chi-square in each regime, whose survival function we evaluate by Imhof's inversion, and AP is the integral of Appendix B.1. The grid was fixed before running, and no trained network or label enters. Figure 3(d) shows this exact map.

The map uses $d = 48$ so that every cycle length divides d; because population AP depends on d, its values are not the references for the trained $d = 64$ models. At $m = 4, \rho = 0.6$ the reversed-graph AP is 0.500, 0.620, and 0.620 for $W = 2, 3, 4$, and the true-graph AP at $W = 3$ is 0.993, the values of the direct integration in Appendix B.1. At $d = 64$ the exact references are 0.639 (reversed, $W \geq 3$) and 0.997 (true), which trained Graph-GRU approaches to within 0.016. In every cell where reversed-graph AP exceeds 0.52, it stays at or below 0.51 for $W < m - 1$ and exceeds 0.52 at $W = m - 1$, as Proposition 2 predicts. The rank correlation between ρ^{m-1} and the $W = 8$ reversed-graph AP over the 20 cells is 1.00, and at $\rho = 0.9$, $m \leq 4$ the reversed graph reaches AP 1.00. An earlier Monte Carlo version of the map, with shared least squares fitted on 4200 simulated normal steps and three simulation seeds, agrees with the exact map to a median absolute difference of 0.001 over the 320 values (largest 0.029, at a cell where the exact AP is at chance); Record Table R8.4 lists its $W = 8$ values.

B.3 Retention beyond Gaussian permutation cycles

Proposition 2 is derived for Gaussian permutation cycles. We repeat the analysis of Appendix B.2 on three other generators with simulation-based references rather than exact population optima, keeping

its interface, shared predictor fitted on normal data, node-averaged squared-residual score, $d = 48$, 4200 normal training steps, 2100 test steps per regime, and three simulation seeds; the grid was fixed before running, and a 500-step burn-in is discarded. (i) Sparse VAR: $x_t = B_y x_{t-1} + \epsilon_t$ with $\epsilon_t \sim \mathcal{N}(0, I)$; each node has $k \in \{1, 2, 3\}$ random parents with weights ± 1 of random sign, B_0 is scaled to spectral radius $r \in \{0.6, 0.9\}$, and $B_1 = B_0^\top$. Supplied graphs carry edge signs and are normalized by absolute row sums; the degree-matched random graph excludes self and the true parents (in the cycles below, also the child). Reversal can change each channel's own dynamics here, so we report self-only AP and the cross-channel share $(\mathrm{AP}_{\mathrm{rev}} - \mathrm{AP}_{\mathrm{self}})/(\mathrm{AP}_{\mathrm{true}} - \mathrm{AP}_{\mathrm{self}})$. (ii) Heavy tails: Equation (2) with Student-t_3 innovations scaled to unit variance. (iii) Nonlinear cycles: $x_t = \rho P_{y_t} \tanh(2x_{t-1})/c + \sqrt{1-\rho^2}\epsilon_t$ with $c = 0.797$, so the drift has unit variance for standard normal input (empirical stationary variance 1.00–1.24); its noiseless drift has slope $2\rho/c > 1$ at zero, so it is bistable. Settings (ii) and (iii) use $m \in \{3, 4\}$, $\rho \in \{0.6, 0.9\}$, and $W = 1, \ldots, 6$; (iii) also fits a shared one-hidden-layer MLP (32 units, early stopping). In (ii) and (iii) each channel's law still depends only on its cycle length, and the target is independent of the supplied child's lags $1, \ldots, m-2$, so Proposition 2's first predictive lag remains $m - 1$ although the predictor is no longer the Gaussian conditional mean.

Table B.1: Simulation-based reference AP (fitted predictor) at the largest window ($W = 4$ for VAR, $W = 6$ otherwise), mean over three seeds, with the reversed-graph min–max over seeds. The last column is the cross-channel share of the reversed graph (VAR) or the first window at which seed-mean reversed-graph AP exceeds 0.52, with $m - 1$ in parentheses (cycles). Figure B.1 shows AP across windows; full grids, per-seed values, and a per-node VAR variant are stored with the analysis code.

Setting	Cell	True	Reversed	Rev. range	Random	Self	Share / first W
VAR	$k = 1, r = 0.6$	0.996	0.520	0.480–0.580	0.494	0.494	0.051
VAR	$k = 1, r = 0.9$	1.000	0.509	0.486–0.521	0.506	0.503	0.013
VAR	$k = 2, r = 0.6$	0.990	0.529	0.504–0.554	0.501	0.501	0.057
VAR	$k = 2, r = 0.9$	1.000	0.504	0.480–0.527	0.496	0.494	0.020
VAR	$k = 3, r = 0.6$	0.994	0.509	0.482–0.550	0.510	0.510	−0.001
VAR	$k = 3, r = 0.9$	1.000	0.528	0.487–0.572	0.489	0.488	0.078
t_3, linear	$m = 3, \rho = 0.6$	0.862	0.664	0.652–0.675	0.509	0.509	2 (2)
t_3, linear	$m = 3, \rho = 0.9$	0.990	0.977	0.956–0.989	0.508	0.508	2 (2)
t_3, linear	$m = 4, \rho = 0.6$	0.858	0.564	0.539–0.583	0.505	0.505	3 (3)
t_3, linear	$m = 4, \rho = 0.9$	0.993	0.968	0.960–0.977	0.503	0.503	3 (3)
tanh, linear	$m = 3, \rho = 0.6$	0.987	0.772	0.767–0.776	0.499	0.499	2 (2)
tanh, linear	$m = 3, \rho = 0.9$	1.000	1.000	1.000–1.000	0.495	0.495	2 (2)
tanh, linear	$m = 4, \rho = 0.6$	0.985	0.599	0.592–0.606	0.507	0.507	3 (3)
tanh, linear	$m = 4, \rho = 0.9$	1.000	0.999	0.999–1.000	0.495	0.495	3 (3)
tanh, MLP	$m = 3, \rho = 0.6$	0.996	0.824	0.816–0.828	0.499	0.499	2 (2)
tanh, MLP	$m = 3, \rho = 0.9$	1.000	1.000	1.000–1.000	0.495	0.494	2 (2)
tanh, MLP	$m = 4, \rho = 0.6$	0.995	0.622	0.609–0.639	0.507	0.507	3 (3)
tanh, MLP	$m = 4, \rho = 0.9$	1.000	1.000	1.000–1.000	0.493	0.493	3 (3)

With heavy tails and the tanh nonlinearity, retention keeps its first lag: for every cell and predictor, seed-mean reversed-graph AP stays at or below 0.52 for $W < m - 1$ and first exceeds it at $W = m - 1$, reaching 0.968–0.977 (t_3) and 0.999–1.000 (tanh) at $\rho = 0.9$, while self-only AP is 0.493–0.512. At $\rho = 0.6$ the level depends on the generator: for $m = 3$ and $m = 4$, reversed-graph AP at $W = 6$ is 0.664 and 0.564 with t_3 innovations and 0.772 and 0.599 (linear) or 0.824 and 0.622 (MLP) in the tanh cycles, against 0.815 and 0.615 in the Gaussian cycles. In the sparse VAR the reversed graph retains little: at $W = 4$ its AP is 0.504–0.529 against true-graph AP of at least 0.990 and self-only AP of 0.488–0.510, and its largest cross-channel share over all windows is 0.207 ($k = 3$, $r = 0.9$, $W = 1$); with a separate predictor per node on unsigned adjacency, reversed-graph AP at $r = 0.9, W = 4$ rises to 0.638–0.734, but degree-matched random graphs reach 0.666–0.725, so this retained information is not specific to reversal.

Three departures from the seed-mean pattern arise from finite samples rather than from the population predictor. With t_3 innovations, 4 of 12 seed–cell curves exceed 0.52 before $W = m - 1$ (at most 0.558), and self-only AP, which reads no graph, reaches 0.548 in the same segments and

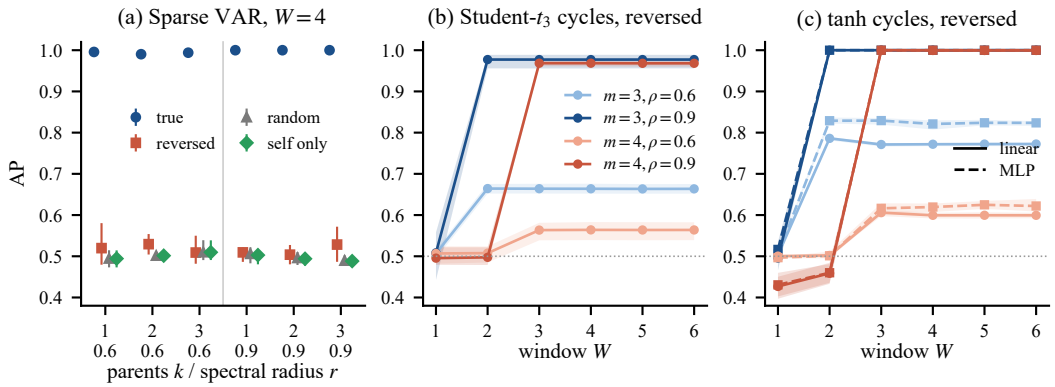


Figure B.1: (a) Sparse VAR at $W = 4$: AP of each supplied graph (markers: seed mean; bars: min–max over seeds). (b, c) Reversed-graph AP against window for heavy-tailed and tanh cycles (bands: min–max over seeds); in (c), solid lines use the linear predictor and dashed lines the MLP. The dotted line marks chance.

windows; one of 12 MLP curves in the tanh cycles does the same (0.535). In the tanh cycles at $m = 4, \rho = 0.9$, reversed-graph AP falls below chance for $W < 3$ with both predictors (0.427–0.460): the fitted child-lag coefficient at $W = 1$ is 0.017–0.033 although its population value is zero, and because that child is the anomalous-regime parent, a small positive weight lowers anomalous residuals.

Trained validation beyond Gaussian cycles. We trained Graph-GRU on the Student-t_3 and tanh four-cycles of Appendix B.3, using the controlled design of the main experiments ($d = 64$, 5000 training steps, alternating 400-step regime blocks, common guard, prevalence 0.5), $\rho \in \{0.6, 0.9\}$, $W \in \{2, 3, 4\}$, true and reversed graphs, and three data seeds; the grid was fixed before running, and the hidden width was reduced from 64 to 32 (40 epochs) to fit the CPU budget. Reversed-graph AP stays at 0.476–0.501 at $W = 2$ and first rises at $W = m - 1 = 3$, to 0.547, 0.950, 0.632, 1.000 against reference values of 0.564, 0.968, 0.616, 1.000 (t_3 at $\rho = 0.6, 0.9$, then tanh); one of 12 seed curves already exceeds 0.52 at $W = 2$ (0.535), and true-graph AP is within 0.017 of its reference value in every cell (Table B.2). These reference values come from Appendix B.3 at $d = 48$. At the trained dimension $d = 64$, linear least-squares references fitted on 20,000 normal steps and scored on 100,000 steps per regime (three simulation seeds, generators unchanged) give reversed-graph AP 0.558, 0.967, 0.622, and 1.000 at $W = 3$ and true-graph AP 0.869, 0.992, 0.994, and 1.000 in the same cell order; trained reversed-graph AP at $W = 3$ lies within 0.017 of these and true-graph AP within 0.022. The MLP reference for tanh is not recomputed at $d = 64$. At $W = 4$ the trained reversed-graph AP falls below the reference value, most strongly for tanh at $\rho = 0.9$ (0.605 against 1.000, and 0.896 against 0.968 for t_3); in the tanh cell the anomalous-regime residual is 0.393 against 2.058 at $W = 3$, so the fitted predictor nearly fits the reversed regime as well. In a post hoc refit of the tanh $\rho = 0.9$, $W = 4$ cell (seed 0, AP 0.596), the original width 64 gives AP 1.000 and width 32 with 120 epochs gives 1.000, and least squares on the same segments gives 1.000; the same refits raise t_3 at $\rho = 0.6$ from 0.516 to 0.551 and 0.555 (reference 0.564), so the $W = 4$ shortfall reflects incomplete optimization of the reduced model rather than missing information.

Table B.2: Trained Graph-GRU AP / simulation-based reference AP for Student-t_3 and tanh four-cycles ($m = 4$). Trained values are means over three data seeds (hidden width 32, 40 epochs); reference values are from Appendix B.3 at $d = 48$ (linear predictor for t_3, MLP for tanh); $d = 64$ linear references are given in the text. Seed ranges, linear reference values for tanh, and a least-squares reference on the trained segments are stored with the analysis code.

		Reversed graph			True graph
Generator	ρ	$W = 2$	$W = 3$	$W = 4$	$W = 3$
t_3	0.6	0.492 / 0.508	0.547 / 0.564	0.509 / 0.564	0.847 / 0.858
t_3	0.9	0.501 / 0.497	0.950 / 0.968	0.896 / 0.968	0.977 / 0.993
tanh	0.6	0.500 / 0.501	0.632 / 0.616	0.581 / 0.619	0.998 / 0.995
tanh	0.9	0.476 / 0.460	1.000 / 1.000	0.605 / 1.000	1.000 / 1.000

C Experimental Setup and Diagnostic Rules

This section collects the settings shared by the main-text experiments and the definitions behind each diagnostic layer. Table C.1 defines the terms used throughout.

Table C.1: **Terms used in the diagnosis.** Tolerance $\tau = 1\%$ and AP tie band 0.005 throughout (Algorithm 1).

Term	Meaning
Supplied graph	The graph passed to the detector, which an ablation replaces.
Executed edges	The message edges the detector's layers actually read; they differ from the supplied graph when the detector learns its own graph.
Native graph	The graph a detector learns or constructs by itself.
Oracle source	A neighbor taken from the known generator, such as a node's normal parent.
Frozen replacement	Weights trained on the original graph; only the test graph is replaced.
Refit	The detector is trained and evaluated on each supplied graph.
Critical / control lag	The lag at which a source predicts the target (lag 1 for a parent, $m-1$ for a child) / another lag of the same source removed with the same operator.
Absorbs	A layer limits the replacement's effect on what it passes on; absorption can be partial.
Information-limited	The original graph's history adds less than τ to self-history on normal data ($g < \tau$).
Information retained	The replacement history keeps a measurable gain ($g_{\mathrm{rep}} \geq \tau$); this does not mean it holds the original information.
Utilization-limited	$g \geq \tau$, but the graph lowers the detector's normal risk against self-only support by less than τ ($e < \tau$).
Graph used	Neither information nor utilization absorbs ($g \geq \tau$, $e \geq \tau$).
Reaches the ranking	The AP contrast is resolved: its interval over replicate units excludes zero, or $\lvert\Delta\mathrm{AP}\rvert > 0.005$ for one run.
Scoring route	AP changes without a measured gain in mean normal risk; predictions and residuals may still differ.
Absorbed at scoring	Normal risk changes but AP does not.
Population reference	AP of the population MSE-optimal forecaster (exact, Gaussian cycles).
Simulation-based reference	AP of a predictor fitted on finite simulated normal data.
Locked	Fixed on normal data before any test score is computed (models, calibrations, normal-data labels).

C.1 Detailed experimental setup

Table C.2 lists the main-text experiments by diagnostic question and replication unit; Table R8.5 extends the map to every experiment, including those retained in the supplementary records.

Controlled cycles. History experiments use 64-channel Gaussian four-cycles with $\rho = 0.6$ and anomaly prevalence 1/2, comparing windows and lags with Graph-GRU (Cho et al., 2014) and

Graph-TCN (Bai et al., 2018). At fixed $W = 4$, the same lag removal is applied to training, validation, and test data before refitting. Ten data seeds are paired, with common guard $g = 8$ for window comparisons (execution replay guard: Record R9.3). Training uses Adam (Kingma & Ba, 2015); Record R13.14 gives full settings. GDN experiments distinguish nominal swaps, executed-support interventions, frozen replacement, and paired refitting; frozen conditions retain normal calibration. Degree-matched controls use five message edges per node and change one source. GDN lag studies use 150-epoch budgets, and the controlled MTGNN test uses the same data seeds, budget, and scoring (Record R9.4).

Real data. The SMD graph ablation uses the 25 objects outside the three historical development objects (Su et al., 2019), with three initializations per object and native learned support. Checkpoints, choices, calibrations, and normal-data verdicts are locked before test scores are computed; scoring outcomes then follow the prespecified rules. The historical study on objects 1-1, 2-1, and 3-1 is reported separately in Record R4.1. The SWaT and PEMS-BAY settings, including the injected replay attacks, are given with their results in Appendix F.4.

Evaluation and statistical units. Only normal training and selection data determine candidates, and decisions are locked after held-out normal evaluation and before action data are opened. Detection results avoid point adjustment and report AP separately from normal MSE, AUROC, and alarm rates. Intervals resample the replicate unit of Table C.2: paired data seeds, and for SMD equally weighted objects after averaging initializations; time-block intervals evaluate fixed-predictor losses. Native contrasts average initializations within data seeds. Initializations do not create new data replicates.

Table C.2: **Main experiments and units of replication.** Initializations do not create new data replicates. Table R8.5 adds detailed controls and candidate validation.

Question / system	Intervention and endpoint	Data units / initializations
Validation / planted	Twelve planted replacements in GDN or MTGNN; blind layer labels against a sealed key	12 cases $\times$ 10 data seeds
Execution / GDN	Supplied swap vs. executed transpose; edges, predictions, score ranks	5 data seeds
History / GRU, TCN	Window and lag removal; normal risk, AP	10 data seeds / fixed init.
Fixed sources / GDN	Parent/child selective-lag refits; risk, AP/AUROC	5 data seeds / 1 init.
Native sources / GDN	Frozen masks and refits; all-node normal risk	5 data seeds / 3 inits.; 20 new seeds / 1 init.
Components / GDN, MTGNN	Single-component replacements on the controlled cycle; AP, exposure, attention	20 data seeds / 1 init. (four arms: 2 inits.)
Real ablation / SMD	Learned, self-only, random, and estimated supports; root weight; MTGNN; ridge gain; normal risk, AP	25 new objects / 3 inits.
Real ablation / SWaT, PEMS-BAY	GDN arms with and without root weight; MTGNN; score aggregation	1 test series each / 3 inits. (SWaT MTGNN: 10)

C.2 Diagnostic layers and experiment map

Table C.3 summarizes what each layer of Algorithm 1 reads and decides; Table R8.5 maps every experiment to these questions.

Table C.3: **What to measure after replacing a graph.** Each layer answers a distinct question. Frozen interventions measure use by a checkpoint; refits measure adaptation.

Layer	Intervention and readout	Interpretation and next check
Execution	Replace supplied graph; record executed edges and predictions.	Unchanged edges locate an interface issue; intervene on actual support.
Information	Compare source-plus-self with self-history risk.	Retained gain motivates lag-specific removal and window controls.
Utilization	Mask exposed lags at a fixed checkpoint; compare refits and history/readout probes.	Separate current use, adaptation, and finite decoding difficulty.
Scoring	Trace paired residuals, calibration, rankings, and thresholded alarms.	Locate where prediction changes cease to improve detection.

D PLANTED-ABSORPTION BENCHMARK

This section gives the design, protocol, and complete results of the benchmark in Section 4, and a report template for graph ablations.

D.1 DESIGN AND PROTOCOL

Data. Every dataset has $d = 32$ channels, each distributed as $\mathcal{N}(0, 1)$ in every regime, apart from the added observation noise of the noise-burst case. *Cycles* follow Equation (2) on directed m-cycles; their anomaly is the reversed regime. *Independent channels* are AR(1) processes with coefficient 0.6; inside an anomaly segment a random quarter of the channels switch to -0.6. The *factor model* is $x_{i,t} = \sqrt{0.6}\,\lambda_i^\top f_t + \sqrt{0.4}\,\epsilon_{i,t}$ with unit loadings $\lambda_i \in \mathbb{R}^2$ and a two-dimensional AR(1) factor with coefficient 0.9; inside a segment a random quarter of the channels follow an independent copy of the factor, which keeps every channel's marginal and autocorrelation. *Noise bursts* add $\mathcal{N}(0, 3^2)$ observation noise to a random quarter of the channels of a cycle. Normal data have 4000 training, 1000 selection, 2048 calibration, and 4096 evaluation steps; each test has 8000 steps with ten 80-step anomaly segments (prevalence 0.10).

Detectors and fitting. All GDN variants are the official model (dimension 64, window 4, top-k 5) with one replaced component, as in Appendix E.4; MTGNN is the official model with its graph constructor. Every arm trains for 100 epochs with Adam (10^{-3}, batch 256) and keeps the checkpoint with minimum normal selection MSE; the checkpoint with minimum selection MSE within the first 30 epochs is kept for the budget sensitivity. Random supports are degree matched and, on cycle data, drawn from other cycles.

Protocol. The protocol, the truth key (by hash), and all code were committed before any formal fit. Ten data seeds were fitted on two machines with disjoint seeds; regenerating the second machine's packets on the first gave bit-identical arrays. Normal-data measurements for all seeds were locked before any sealed test was opened. The diagnosis stage applies the unchanged rules of Algorithm 1 and never opens the truth key, which is compared only afterwards. The primary setting is the ridge estimator, the full budget, $\tau = 1\%$, and a tie band that treats $|\Delta\mathrm{AP}| \le 0.005$ as unchanged; per-seed AP intervals resample the ten 800-step test blocks, and pooled intervals are paired t intervals over seeds. A two-epoch development run on a separate seed checked only that every stage runs.

D.2 REPORT TEMPLATE

Table D.2 lists what a graph ablation can report so that a reader can locate its outcome. Every row except the self-only fit reuses outputs that an ablation study already produces.

D.3 RESULTS

Table D.3 gives every case. The information-layer labels are identical for the ridge, blocked cross-validated ridge, and gradient-boosting estimators and for every $\tau \in \{0.5, 1, 2, 5\}\%$, so exact-label accuracy is 84% in every setting with the full budget and the tie band, 88% with the 30-epoch budget,

Table D.1: **Planted cases.** Each case replaces the original support by the replacement support and refits. The planted component determines the layer that should absorb the replacement.

Case	Data	Detector	Original → replacement; planted component	Expected label
P01	4-cycle	GDN, native	supplied true → reversed; top-k graph ignores the supplied graph	execution
P02	4-cycle	GDN (GATv2), symmetrized	supplied true → reversed; both symmetrize to {parent, child}	execution
P03	4-cycle	MTGNN	supplied true → reversed; learned constructor ignores it	execution
P04	independent	GDN (GATv2)	random → random; no cross-channel information	information-limited
P05	8-cycle	GDN (GATv2)	child → random; the child predicts only at lag $7 > W$	information-limited
P06	4-cycle, ρ=0.9	GDN (GATv2)	parent → child; lag-3 history retains the information	information retained
P07	factor	GDN (GATv2)	best loadings → random; every channel carries the factor	information retained
P08	4-cycle	GDN native → fixed	learned → random executed edges; top-k selector without direct gradient	utilization
P09	4-cycle	GDN (static)	parent + 23 distractors → 24 distractors; static attention dilutes	utilization
P10	4-cycle + bursts	GDN (GATv2)	parent → random; bursts are ranked perfectly without the graph	scoring
P11	4-cycle	GDN (GATv2)	parent → random; nothing absorbs	reaches ranking
P12	8-cycle	GDN (GATv2)	parent → child; nothing absorbs	reaches ranking

Table D.2: **A report template for graph ablations.**

Layer	Report	Cost
Execution	executed edges and maximum prediction change when the supplied graph is swapped in a frozen checkpoint	one forward pass
Information	ridge gain of the original and replacement supports over self-history on normal data, with τ	ridge fits
Utilization	normal MSE with the graph and with self-loops only, same detector and budget	one extra fit
Scoring	AP with prevalence, AUROC, and alarm rates at the calibrated threshold, for each support	saved scores
Protocol	frozen or refitted replacement; number of initializations and data seeds	–

and 76% (full) and 79% (30 epochs) when an AP change counts whenever its interval excludes zero. All misses are in P08 and P09. In P08 the utilization layer flags native GDN's graph in 9 of 10 seeds ($e < 1\%$), but the replacement lowers AP by 0.041 because native GDN still exposes some parents, so the rule reports a scoring route; in P09 static attention extracts a normal-risk gain of 1.3%, above $\tau = 1\%$, and the ranking changes by 0.013. At $\tau \geq 2\%$ the utilization layer flags P09 as well, and the label again becomes a scoring route. The pre-registered labels are reported unchanged.

Existing importance readouts. To compare the rules with the readouts that graph-importance studies use, we computed three more readouts on every locked original checkpoint, on normal data only, under a protocol committed before the run: permutation importance (the other channels' windows taken jointly from a permutation of time), conditional randomization importance (the other channels' windows drawn from their Gaussian conditional given the target's own window, a Gaussian analogue of conditional permutation importance (Strobl et al., 2008)), and GraphMask (Schlichtkrull et al., 2021) in its non-amortized form (hard-concrete edge gates, gated-off messages replaced by a learned baseline, the fewest kept edges whose normal MSE rises by at most

Table D.3: **Planted-absorption benchmark: measurements, baseline readouts, and labels** (means over ten data seeds; primary setting). g, g_{rep}: ridge gain (%) of the original and replacement supports over self-history; e: normal-risk gain (%) of the original arm over self-only support; AP of the original and replacement arms at prevalence 0.10 and their difference; occlusion: normal-MSE increase (%) when all non-self messages of the frozen original are removed; attention: weight on non-self sources. Labels: X execution, I information-limited, R retained, U utilization, S scoring, SR scoring route, N reaches ranking. Occlusion and attention are not defined for MTGNN (P03).

Case	g	g_{rep}	e	AP orig.	AP repl.	ΔAP	Occlusion	Attention	Planted	Pooled	Seeds
P01	34.9	2.9	0.4	0.144	0.153	-0.009	3.6	0.79	X	X	10/10
P02	34.9	2.9	34.8	0.700	0.700	-0.000	53.6	0.96	X	X	10/10
P03	34.9	2.9	11.3	0.365	0.339	0.026	–	–	X	X	10/10
P04	-0.4	-0.4	-0.1	0.892	0.892	0.000	-0.0	0.21	I	I	10/10
P05	-0.0	-0.0	-0.0	0.103	0.103	0.001	0.0	0.49	I	I	10/10
P06	66.6	17.6	66.5	0.993	0.949	0.044	89.4	0.96	R	R	10/10
P07	11.9	8.3	8.8	0.241	0.137	0.103	23.4	0.68	R	R	10/10
P08	32.9	-0.4	0.4	0.144	0.103	0.041	3.6	0.79	U	SR	1/10
P09	33.4	-1.1	1.3	0.116	0.103	0.013	40.5	0.96	U	N	0/10
P10	34.9	-0.1	34.8	0.991	0.991	0.000	52.5	0.94	S	S	10/10
P11	34.9	-0.1	34.8	0.812	0.103	0.709	52.5	0.94	N	N	10/10
P12	35.8	-0.0	35.8	0.815	0.103	0.712	52.4	0.94	N	N	10/10

1%). Two structural facts limit any readout of the original model: P01 and P08 share one original model, and so do P10 and P11, although their planted layers differ. We then gave every readout the planted labels and trained a nearest-centroid classifier of the planted layer (Table D.4). On a held-out case, no readout names the absorbing layer in more than 51% of runs, and all six together reach 27%; even with every case seen in training, all six reach 75%. The blind rules, which use no labels, name the layer in 84% of runs. The first seed's readouts were computed twice during development, and the constraint repair of GraphMask was added after the first attempt violated the constraint; the formal run recomputed all seeds with the committed code.

Table D.4: **Layer accuracy (%) of existing importance readouts on the planted benchmark** (120 runs). Each readout, alone or all six together, trains a nearest-centroid classifier of the planted layer with the planted labels; leave-one-case-out holds out a whole case (an unseen absorber), leave-one-seed-out holds out a data seed (every case seen). The blind rules of Algorithm 1 use no labels and name the layer in 84% of runs.

Readout	Leave one case out	Leave one seed out
Refitted ablation ΔAP	32	47
Occlusion of non-self messages	20	31
Attention on non-self sources	51	52
Permutation importance	32	44
Conditional randomization importance	30	38
GraphMask (kept edges)	29	46
All six readouts	27	75
Layered diagnosis (blind rules)	84	

E Complete Detectors: Execution, Lag Use, and Graph Selection

This section audits what a supplied graph does in public detectors, reruns GraGOD's released pipeline, and reports the selective-lag study and the single-component replacements of Section 5; further controls are in Record R13.

E.1 Supplied graphs in public graph detectors

The execution layer asks whether a supplied graph reaches message passing. Table E.1 answers this for seven public implementations of learned-graph detectors, read at pinned commits; for GDN

and MTGNN the answer is also measured by swapping the supplied graph of trained checkpoints (Section 6 and Appendix F.2). Whenever the graph is learned, a supplied graph either never enters message passing or only initializes a trained adjacency, so a supplied-graph ablation of these detectors tests at most an initialization. The selector column connects this audit to Record R9.4: the three GDN-type implementations rank sources by a symmetric cosine similarity.

Table E.1: **Role of a supplied graph when the detector learns its graph.** Official code unless noted; commits and code locations are recorded with the audit.

Implementation	Supplied graph when the graph is learned	Graph selector
GDN (Deng & Hooi, 2021)	Ignored; measured (no executed edge changes on 75 SMD checkpoints)	Top-k cosine similarity (symmetric)
FuSAGNet (Han & Woo, 2022)	Ignored: the edge-index argument of the forward pass is unused	Top-k cosine similarity (symmetric)
GDN in GraGOD (Bello et al., 2026)	Ignored with `learn_graph`, which every released configuration enables; executed without it	Top-k cosine similarity (symmetric)
MTGNN (Wu et al., 2020)	Ignored with graph learning; measured (prediction difference 0)	Antisymmetric scores, top-k (directed)
GANF (Dai & Chen, 2022)	Initializes a trained adjacency	Learned acyclic graph (directed)
GTA (Chen et al., 2022)	No supplied-graph input; gates an internal complete graph	Gumbel-softmax gates (directed)
GReLeN (Zhang et al., 2022)	No supplied-graph input; relations inferred per window	Gumbel-softmax relations (directed)

Graph learning in GraGOD's released configurations. In GraGOD's GDN, the `learn_graph` argument of the model constructor (`models/gdn/model.py`) defaults to true, and the training script passes each configuration's model parameters to the constructor unchanged. All four released GDN configurations therefore learn the graph: `params_swat.yaml` and `params_ute.yaml` set it to true, and `params.yaml` and `params_telco.yaml` inherit the default. The only released configuration that supplies a random graph, `params_ute.yaml` (edge index `edge_index_random.pt`), thus runs with graph learning on, and in that setting the supplied random graph is not executed. The published topology comparison reports GDN on SWaT and TELCO (Bello et al., 2026); the repository does not include the configurations of those runs, so whether they disabled graph learning cannot be determined from the code.

E.2 Executing a supplied graph in GraGOD's released pipeline

The audit shows that GraGOD's released GDN configurations do not execute a supplied graph. We therefore ran its detectors with graph learning on and off, and while preparing the run we found a second execution-layer absorber. In the training and prediction steps of GraGOD's GDN (branch `develop`, commit `ec8cd45`; the same lines are in the `anonimized` branch), the (batch, window, sensor) tensor from the sliding-window dataset is reshaped to (batch, sensor, window) with `reshape`, not transposed, and GraGOD's GCN reshapes the same tensor to (batch·sensor, window). A probe tensor whose entries encode time and sensor shows that under this layout none of the 51 SWaT nodes receives its own sensor's history: node 0 receives sensors 0–4 at the oldest time step, and every node receives five different sensors. The supplied graph is then executed, but on node inputs that do not correspond to the sensors it connects.

Design. The protocol, the vendored model code, and the runner were committed before any formal fit. We use the SWaT packet of Appendix F.4 (one-minute resolution, 51 sensors), GraGOD's GDN with its released SWaT parameters (window 5, top-k 15, embedding 64) and its GCN (window 10, three TAGConv layers), GraGOD's training schedule (early stopping on validation loss, at most 200 epochs), and ten initializations. We cross the released layout with a transposed one, GDN's

`learn_graph` on and off, and four supplied graphs: GraGOD's SWaT topology rule (sensors whose names differ only in the last character), a degree-matched random graph, the complete graph, and no edges. Scores follow GraGOD (absolute errors standardized by the median and IQR of the scored series, 5-point mean, channel mean); calibrated channel-max scores are secondary. All normal-data records were locked before the test was scored once. Initialization 0 ran on the local GPU and initializations 1–9 on the remote GPU; every contrast is within one initialization and one machine.

Table E.2: **GraGOD's GDN and GCN on SWaT under the released and a transposed input layout** (ten initializations; means). Learn: GDN's `learn_graph`. Graph: the supplied graph (GraGOD's SWaT topology rule, a degree-matched random graph, the complete graph, or no edges). Normal MSE on the normal evaluation split. AP with GraGOD's score (median/IQR of the scored series, 5-point mean, channel mean) and with calibrated channel-max scores; prevalence 0.125.

Detector	Layout	Learn	Graph	Normal MSE	AP (GraGOD)	AP (max)	AUROC
GDN	released	on	topology	0.878	0.732	0.609	0.834
GDN	released	on	random	0.875	0.728	0.534	0.832
GDN	released	off	topology	0.595	0.754	0.726	0.860
GDN	released	off	random	0.938	0.744	0.675	0.856
GDN	released	off	complete	0.714	0.717	0.615	0.827
GDN	released	off	self only	1.062	0.732	0.698	0.849
GDN	transposed	on	topology	0.883	0.725	0.594	0.846
GDN	transposed	on	random	0.888	0.733	0.547	0.848
GDN	transposed	off	topology	0.619	0.728	0.635	0.825
GDN	transposed	off	random	0.221	0.740	0.738	0.854
GDN	transposed	off	complete	1.321	0.502	0.578	0.792
GDN	transposed	off	self only	0.201	0.717	0.509	0.833
GCN	released	off	topology	0.892	0.733	0.720	0.858
GCN	released	off	random	0.777	0.741	0.723	0.858
GCN	released	off	self only	1.003	0.758	0.612	0.862
GCN	transposed	off	topology	0.153	0.728	0.734	0.830
GCN	transposed	off	random	0.179	0.747	0.739	0.848
GCN	transposed	off	self only	0.183	0.775	0.410	0.853

Table E.3: **Paired AP contrasts on SWaT** (GraGOD score; mean over ten initializations with paired t 95% interval; wins/ties/losses with tie band 0.005).

Contrast	ΔAP	W/T/L
GDN, released, learn on: topology − random	$+0.004\,[-0.003, +0.012]$	4/5/1
GDN, released, learn off: topology − random	$+0.010\,[-0.006, +0.025]$	6/3/1
GDN, released, learn off: topology − self only	$+0.022\,[+0.008, +0.037]$	8/2/0
GDN, transposed, learn off: topology − random	$-0.012\,[-0.030, +0.005]$	1/4/5
GDN, transposed, learn off: topology − self only	$+0.010\,[-0.006, +0.027]$	5/4/1
GDN, transposed, learn off: complete − self only	$-0.215\,[-0.378, -0.053]$	1/0/9
GCN, released: topology − random	$-0.007\,[-0.022, +0.007]$	2/3/5
GCN, transposed: topology − random	$-0.019\,[-0.037, -0.001]$	1/1/8
GCN, transposed: topology − self only	$-0.047\,[-0.058, -0.035]$	0/0/10

Layer by layer. With graph learning on, replacing the supplied graph of a trained model changes its predictions by at most 2.9×10^{-6} on every initialization, and the refitted topology-minus-random contrast is $+0.004\,[-0.003, +0.012]$: the replacement is absorbed at execution. Under the released layout, self-only GDN cannot forecast (normal MSE 1.06 for unit-variance data), whereas the transposed layout reaches 0.20. At the information layer, the topology graph's sources add 16.0% to self-history and a degree-matched random graph keeps a smaller gain, so the diagnosis labels the topology-to-random replacement information retained in every executed configuration. At the utilization layer, transposed GDN with the topology graph predicts normal data worse than with self-loops (0.62 against 0.20), as on SMD. At the scoring layer, AP stays between 0.717 and 0.775 in 17 of the 18 configurations while their normal MSE ranges from 0.15 to 1.32; over all 180 fits except the transposed complete-graph GDN (AP 0.502), the Spearman

correlation of normal MSE with AP is -0.04. The topology graph beats self-only support under the released layout ($+0.022\,[+0.008, +0.037]$), where self-only nodes lack their own history, but not under the transposed layout ($+0.010\,[-0.006, +0.027]$); for transposed GCN it lowers AP ($-0.047\,[-0.058, -0.035]$). GDN's topology-minus-random contrast is unresolved in every configuration (Table E.3). This study uses our one-minute SWaT packet rather than GraGOD's own preprocessing, and it does not re-evaluate the published numbers; it shows what the released code executes and where a graph replacement stops acting on these data.

E.3 Selective history removal inside complete GDN

The source comparisons above change whole neighbor histories. Here we test the predicted lag mechanism inside the complete detector while keeping source identities and message counts fixed. The intervention removes one entry from the designated source history, and the full network is refitted under that restriction. This connects the analytical information comparison to a trained GDN and its anomaly scores.

Matched source histories. We generate 64-channel Gaussian four-cycles with $\rho = 0.6$ and window $W = 4$. Every target receives five messages: self, its normal parent or normal child, and three distinct nodes outside its cycle. The three cross-block sources are identical between parent and child arms and across all lag conditions within a data seed. They are sampled once and held fixed. Parent and child are oracle sources defined by the generator. Full history, removal of designated-source lag 1, and removal of designated-source lag 3 give six conditions per seed. Self-history and all cross-block histories remain intact.

The lag mask acts on the raw source window for the designated non-self edge before its linear projection. Both its attention key and message value use the masked window; the target query and self-loop retain the full window. Attention normalization, aggregation, embedding multiplication, batch normalization, dropout, and the original output head are retained. Masks apply during fitting, selection, calibration, and evaluation. The adapter adds no trainable parameters: all conditions have 4993 parameters. Training retains the original batch-normalization behavior; evaluation uses stored running statistics. The original vendor source is unchanged.

Data roles, the fixed training schedule, the pre-run execution checks, and the calibration order are reported in Record R13.

Available information and realized gains. For the parent arm, full history and lag-3 removal retain the lag-1 predictor, with population risk $1-\rho^2 = 0.64$. For the child arm, full history and lag-1 removal retain lag 3, with risk $1-\rho^6 = 0.953344$. Removing the critical lag in either arm leaves the target's own lag 4, with risk $1-\rho^8 = 0.983204$. Thus the critical-minus-control population risk contrasts are 0.343204 for parent and 0.029860 for child. These are conditional-information references; the neural predictor need not attain them.

Table E.4 shows complementary lag responses in the trained detector. Parent GDN risk rises from 0.7685 to 0.9912 after lag-1 removal; its lag-3 control has risk 0.7665. Child risk rises from 0.9685 to 0.9913 after lag-3 removal; its lag-1 control has risk 0.9679. The prespecified critical-minus-control contrasts are $0.22474\,[0.21789, 0.23281]$ and $0.02348\,[0.02287, 0.02410]$, respectively, and are positive in every data seed. Brackets are descriptive paired-seed 95% percentile intervals from 5000 bootstrap resamples, without multiplicity correction; five data draws provide limited precision. Executed-history probes closely follow the analytical references, while full-history GDN risks remain higher. The experiment therefore identifies both the lag-dependent benefit realized by the full model and predictive information it leaves unused.

Detection and training-budget responses. The same interventions change detection: parent AP falls from 0.9729 to 0.5050 after lag-1 removal, and child AP from 0.5683 to 0.5053 after lag-3 removal. The opposite-lag controls retain AP 0.9729 and 0.5698. Critical-minus-control AP contrasts are $-0.46788\,[-0.48047, -0.45546]$ and $-0.06442\,[-0.07419, -0.05584]$. The complementary directions already appear at 30 and 75 epochs, with a larger child-source benefit at longer budgets (Table R3.10). Four of 30 best checkpoints occur at epoch 150, so these fixed-budget comparisons do not establish convergence. Common switch exclusions preserve the pattern. These results trace

Table E.4: **Selective history removal inside complete GDN.** Five paired data seeds; 30 refits at $W = 4$. Each node retains self, the parent or child, and three common cross-block sources. Only the designated source's named lag is masked. MSE uses independent normal trajectories; detection retains internal regime transitions. Tables R3.9 and R3.10 give individual seeds and shorter budgets.

Source	History	GDN MSE	History MSE	AP	AUROC
Parent	Full	0.7685	0.6423	0.9729	0.9734
	Drop lag 1	0.9912	0.9849	0.5050	0.4993
	Drop lag 3	0.7665	0.6422	0.9729	0.9738
Child	Full	0.9685	0.9553	0.5683	0.5742
	Drop lag 1	0.9679	0.9551	0.5698	0.5766
	Drop lag 3	0.9913	0.9850	0.5053	0.4985

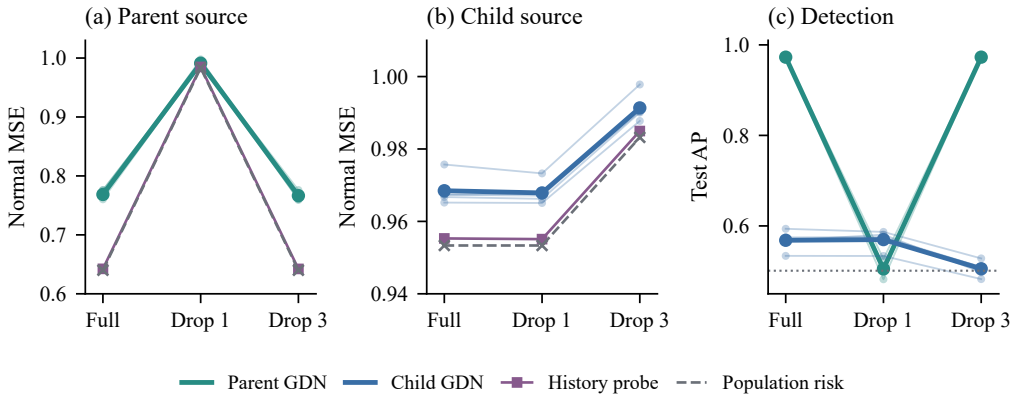


Figure E.1: **Detailed selective-lag responses with fixed oracle sources.** Thin lines are five paired data seeds; thick lines are means. History probes and population risks separate accessible information from GDN's realized gains. Risk panels use different vertical scales; the dotted AP line is prevalence. All fits use 150 epochs, normal-only selection/calibration, and retain internal test transitions.

the predicted source lag through full-network fitting and normal calibration into anomaly ranking under the specified oracle supports.

The two lag-level studies of GDN with its own learned graph (native-support lag masks over five data seeds with three initializations, and their frozen confirmation on twenty new data seeds) are reported in Record R10.

E.4 Single-component replacements in GDN on the controlled cycle

This study locates native GDN's near-chance detection of the cycle reversal (Section 5.2) by replacing one component at a time. Its protocol, runners, and analysis script were committed before any formal fit; an addendum with the two combined arms was committed after the first normal-data records and before any test score was read.

Exposure against random choice. From the executed edges stored with the 20-seed native GDN study, we count the targets whose sources include their parent, child, or antipode and compare with a relabeling null that keeps each learned graph's structure and permutes node labels (20000 relabelings per graph; its mean equals the closed form $64 \cdot 4/63 = 4.06$). Parent exposure is 4.25 against 4.07 (paired t over seeds, $p = 0.59$; 7 seeds above, 13 below), the child is exposed more often than chance (5.50, $p = 0.013$), the antipode less (2.55, $p = 0.003$), and within-cycle edges make up 4.80% of GDN's non-self edges against 4.76% for random choice. The five-seed study gives 3.67, 3.93, and 3.40, none different from random. The learned graph thus does not concentrate on the cycle, and its small departures from random do not favor the parent. For MTGNN's learned graph the parent

is exposed in the direction its first propagation reads on 12.7 targets against 3.98 at random (all 20 seeds above), and on 18.7 in either direction against 7.96; counted either way, parent and child exposure coincide by construction, so only the one-direction count distinguishes them.

Static attention on a supplied cycle. GDN's attention logit for target i and source j is $e_{ij} = \mathrm{LeakyReLU}(u_i + s_j)$, where $u_i = a_t^\top [Wx_i \,\|\, v_i]$ depends only on the target and $s_j = a_s^\top [Wx_j \,\|\, v_j]$ only on the source, with v the node embeddings (Brody et al., 2022).

Proposition 4 (Static attention on a supplied cycle). Let every target of a four-cycle C receive all four nodes of C, itself included. Then in every window all targets of C order their attention weights over C identically, and at most one target of C gives its parent a strictly largest weight.

Proof. LeakyReLU is strictly increasing and the softmax preserves order, so for each target the weights over C are ordered by s_j alone. If s has no unique maximum on C, no target gives any source a strictly largest weight. Otherwise let x be the maximizer: every target gives x the largest weight, so a target gives its parent the largest weight only if its parent is x, which holds for exactly one node of the cycle, the child of x. □

With the fixed cycle support, the measured fraction of windows in which all four targets of a cycle share one ranking is 1.00 under static attention and 0.00 under GATv2, whose logit $a^\top \mathrm{LeakyReLU}(W_r[Wx_i \| v_i] + W_l[Wx_j \| v_j])$ does not separate into target and source terms.

The arms, budget, and pre-run equality checks are specified in Record R13.

Results. Tables E.5 and E.6 give every arm and contrast. The rerun of native GDN reaches AP 0.545 on the same packets as the earlier twenty-seed study (0.515); both are near chance. The two runs share the data, the batch orders, and the initialization seed, but GPU scatter operations are not deterministic, so the training trajectories diverge (the best epoch differs in every seed). Near-chance native GDN is sensitive to this: per-seed AP differences between the runs have standard deviation 0.050, five seeds differ by 0.09–0.14, and the mean difference is the same on the two machines that fitted the seeds (0.031 and 0.029). We therefore read the gap as run-to-run variability of a selector without direct gradient, not as a difference in setup. A trainable selector finds the cycle and raises AP to 0.951–0.952 on every seed, whether its score is symmetric or directed ($+0.001\,[-0.002, +0.004]$). Given the cycle as sources, static attention reaches 0.857 and GATv2 0.981 ($+0.124$, every seed): static attention splits its weight 0.33, 0.17, and 0.29 between parent, child, and antipode and ranks the candidates of a cycle identically in every window checked (1024 per seed), whereas GATv2 puts 0.95 on the parent and never shares one ranking across a cycle. With trainable selectors, GATv2 adds $+0.016$ (directed) and $+0.017$ (symmetric), and the two selectors stay equal ($-0.001\,[-0.007, +0.005]$). A supplied oracle parent with static attention reaches only 0.885, because static attention keeps 0.43 of the weight on the node itself, and learned sources stay 0.014 below the supplied cycle under GATv2. In MTGNN the symmetric constructor detects better than the official antisymmetric one ($-0.096\,[-0.148, -0.044]$; 13 of 20 seeds). At prevalence 0.05 every pre-registered contrast keeps its call. The pre-registered interpretation is thus: C2 positive and C3 positive, static attention loses direction once the right sources are supplied and GATv2 recovers it; C1 null, selector symmetry does not explain native GDN's near-chance detection, which a trainable choice removes; C4 negative, MTGNN's detection does not depend on its antisymmetric score.

Second initialization. Four arms were repeated with initialization 1 on the same 20 packets, registered before any test of this study was read. AP is 0.509 (native), 0.854 (cycle sources, static), 0.981 (cycle sources, GATv2), and 0.966 (directed selector with GATv2), against 0.545, 0.857, 0.981, and 0.967 with initialization 0. With initialization 1 the contrasts C2, C3, and C7 are $+0.127\,[+0.121, +0.134]$, $+0.345\,[+0.337, +0.352]$, and $+0.458\,[+0.450, +0.465]$, positive on every seed, and every seed's contrast has the same sign under both initializations.

Oracle-parent AP across studies. GDN with an oracle parent reaches different AP in different studies because the other executed sources, the data seeds, and in some studies the window and training budget differ. Here each target reads only itself and its parent ($W = 4$, 150 epochs), and static attention keeps 0.43 of the weight off the parent (AP 0.885). In the selective-lag study (Appendix E.3; same window and budget, five other data seeds), each target also reads three fixed

cross-block sources (AP 0.973). The fixed-budget study (Record R13.8) uses the same three extra sources with $W = 16$ and 30 epochs (AP 0.959 over ten seeds), and the capable-checkpoint swaps in the same appendix give 0.889 and 0.968 in two five-seed batches (Record R13.8). All of these lie far above native GDN; we do not attribute their spread to a single factor.

Table E.5: **Single-component replacements on the directed four-cycle** (20 data seeds, initialization 0, prevalence 0.501). AP with seed-bootstrap 95% interval; AP at prevalence 0.05 by reweighting each seed's ROC; normal-evaluation MSE (population optimum $1 - \rho^2 = 0.64$); exposure: targets (of 64) whose executed sources include their parent or child (random choice of four sources: 4.06), from this study's fits, so native GDN's parent exposure (4.5) differs from the 4.25 counted on the earlier twenty-seed study's executed edges; attention: mean weight on the parent.

Arm	Seeds	AP	AUROC	AP ($\pi = 0.05$)	Normal MSE	Parent exp.	Child exp.	Parent att.
GDN, native	20	0.545 [0.525, 0.567]	0.539	0.069	0.986	4.5	4.2	0.03
Symmetric selector	20	0.951 [0.948, 0.954]	0.951	0.654	0.790	39.5	38.1	0.47
Directed selector	20	0.952 [0.949, 0.955]	0.953	0.647	0.761	46.1	12.6	0.52
Cycle sources, static	20	0.857 [0.850, 0.863]	0.855	0.361	0.829	64.0	64.0	0.33
Cycle sources, GATv2	20	0.981 [0.980, 0.983]	0.982	0.811	0.641	64.0	64.0	0.95
Symmetric selector, GATv2	20	0.968 [0.962, 0.973]	0.969	0.738	0.716	47.4	47.2	0.69
Directed selector, GATv2	20	0.967 [0.965, 0.970]	0.968	0.733	0.737	44.1	13.5	0.63
Oracle parent, static	20	0.885 [0.880, 0.890]	0.897	0.379	0.752	64.0	0.0	0.57
MTGNN, symmetric constructor	20	0.943 [0.932, 0.952]	0.943	0.623	0.796	–	–	–
MTGNN, official	20	0.847 [0.797, 0.896]	0.843	0.436	0.831	–	–	–

Table E.6: **Paired contrasts over data seeds** (pre-registered primary contrasts C1–C4 and C7–C9; others secondary). Mean AP difference with paired t 95% interval, wins/ties/losses (tie $|\Delta\text{AP}| \leq 0.005$), exact sign-test p, and the same difference at prevalence 0.05.

Contrast	Seeds	ΔAP	W/T/L	Sign p	ΔAP ($\pi = 0.05$)
Directed − symmetric selector (static)	20	$+0.001\,[-0.002, +0.004]$	4/12/4	0.82	$-0.007\,[-0.023, +0.009]$
GATv2 − static attention (cycle sources)	20	$+0.124\,[+0.118, +0.131]$	20/0/0	1.9×10^{-6}	$+0.449\,[+0.432, +0.467]$
Cycle sources − native (static)	20	$+0.312\,[+0.290, +0.333]$	20/0/0	1.9×10^{-6}	$+0.293\,[+0.271, +0.315]$
MTGNN official − symmetric constructor	20	$-0.096\,[-0.148, -0.044]$	6/1/13	0.12	$-0.187\,[-0.302, -0.073]$
Directed selector + GATv2 − native	20	$+0.422\,[+0.400, +0.444]$	20/0/0	1.9×10^{-6}	$+0.665\,[+0.646, +0.683]$
Directed − symmetric selector (GATv2)	20	$-0.001\,[-0.007, +0.005]$	6/6/8	0.82	$-0.005\,[-0.039, +0.029]$
GATv2 − static (directed selector)	20	$+0.016\,[+0.013, +0.018]$	20/0/0	1.9×10^{-6}	$+0.086\,[+0.069, +0.104]$
GATv2 − static (symmetric selector)	20	$+0.017\,[+0.011, +0.023]$	17/2/1	4.0×10^{-5}	$+0.084\,[+0.048, +0.119]$
Oracle parent − cycle sources (static)	20	$+0.029\,[+0.022, +0.035]$	19/1/0	4.0×10^{-5}	$+0.017\,[-0.001, +0.035]$

F Real-Data Studies: SMD, SWaT, and PEMS-BAY

This section documents Section 6: the layered graph ablation on the 25 SMD objects outside the historical development objects, the follow-up analyses of the learned graph, the information-layer robustness and root-weight repair, and the diagnosis on SWaT and PEMS-BAY. A preliminary study of predictor adoption on the same objects, whose test labels the graph ablation reuses, is retained in Record R9.5.

F.1 Layered graph ablation on 25 SMD objects

This study asks where the null result of a real graph ablation comes from. It uses the same 25 SMD objects, normal and labeled test segments, and native learned-support GDN checkpoints (three initializations per object) as Record R9.5; historical objects 1-1, 2-1, and 3-1 are excluded. The protocol, runner, and analysis were committed before the formal run. Development used only historical object 1-1 with two-epoch fits; it showed the 608-feature all-channel ridge overfitting with penalties up to 100, and the penalty grid was widened to $\{0.01, 1, 100, 10^3, 10^4\}$ before any of the 25 objects was touched.

Supports and fitting. Each object and initialization has five supports: the existing learned checkpoint (no retraining); self-only support (self-loops only); degree-matched random support (four random sources plus self, the construction of Record R4.1); and learned and self-only supports retrained with a longer budget (up to 200 epochs, minimum 30, patience 30) as a baseline-strength check.

Standard fits use the learned protocol (cap 60, minimum 30, patience 12). Every support has its own normal calibration and the historical IQR/max score; no point adjustment is used.

Information layer. Before any detector is trained and without labels, per-node ridge predictors on normal training data use either the node's own 16 lags or all 38×16 channel lags; penalties are chosen on normal selection data and risks are reported on normal evaluation data. The cross-channel gain is $g = 1 - \mathrm{MSE}_{\mathrm{all}}/\mathrm{MSE}_{\mathrm{self}}$.

Endpoints and layer labels. The primary endpoint is the Spearman correlation across objects between g and the normal-risk graph effect $e = 1 - \mathrm{MSE}_{\mathrm{learned}}/\mathrm{MSE}_{\mathrm{self}}$, with a 95% object-bootstrap interval and a one-sided permutation test. Secondary endpoints are the same correlation with the long-training arms and with AP differences, object-averaged AP differences with paired-object bootstrap intervals, baseline-strength ratios, and execution records. Before test labels were opened, each object received one label: information-limited if $g < 1\%$; utilization-limited if $g \geq 1\%$ but the long-training learned graph does not lower normal MSE by at least 1% relative to long-training self-only support; and graph used otherwise. SMD test labels had already been opened in the preliminary adoption study (Record R9.5), so this is a reuse of the official test sets rather than an independent test.

Results. All 375 fits and the verification completed in one formal run. The primary endpoint is Spearman 0.35 between g and e, with 95% object-bootstrap interval $[-0.09, 0.69]$ and one-sided permutation $p = 0.045$; because this interval, like that of the self-only AP difference below, crosses zero, the main text uses both as descriptive summaries. The secondary correlations of g with the long-training effect, with AP(learned) minus AP(self-only), and with AP(learned) minus AP(random) are $0.34\,[-0.07, 0.66]$ ($p = 0.050$), $0.25\,[-0.20, 0.62]$ ($p = 0.107$), and $0.18\,[-0.30, 0.61]$ ($p = 0.184$). The learned graph raises normal MSE over self-only support on all 25 objects under both budgets (e median -0.72; long-training median -0.69) and over random support on 22 of 25; among the 14 utilization-limited objects the median of e is -0.56. Of the 11 information-limited objects, 9 have negative g (minimum -34.7%), reflecting the estimation cost of the 608-input ridge. The longer budget lowers learned-graph MSE by a median of 4% and self-only MSE by 1%; self-only GDN matches the self-history ridge (median ratio 1.00), whereas learned-graph GDN has 1.71 times its normal MSE. Object-averaged AP is 0.394 (learned), 0.346 (self-only), and 0.399 (random). AP(learned) minus AP(self-only) is $+0.048\,[-0.005, +0.116]$ with 14 wins, 2 ties, and 9 losses ($+0.053\,[-0.003, +0.122]$, 16/1/8, under the long budget); AP(learned) minus AP(random) is $-0.005\,[-0.026, +0.015]$ with 11/1/13. Layer labels assign 11 objects to information-limited, 14 to utilization-limited, and none to graph used; their mean learned-minus-self-only AP differences are $+0.028$ and $+0.064$. Across the 75 learned checkpoints the supplied-graph swap changes no executed edge (largest prediction difference 1.9×10^{-6}); the frozen executed transpose lowers AP by $0.016\,[0.002, 0.033]$ with mean score rank correlation 0.83. Table R4.13 lists every object and Figure F.1 shows both layers.

Table F.1: **Graph ablation of GDN on 25 SMD objects.** Object means after averaging three initializations; each support has its own normal calibration and the historical IQR/max score; mean prevalence 0.041. Long training: up to 200 epochs, patience 30.

Support	Normal MSE	AP	AUROC
Learned (native GDN)	0.368	0.394	0.811
Self-only	0.194	0.346	0.821
Degree-matched random	0.273	0.399	0.816
Learned, long training	0.362	0.390	0.818
Self-only, long training	0.191	0.338	0.821

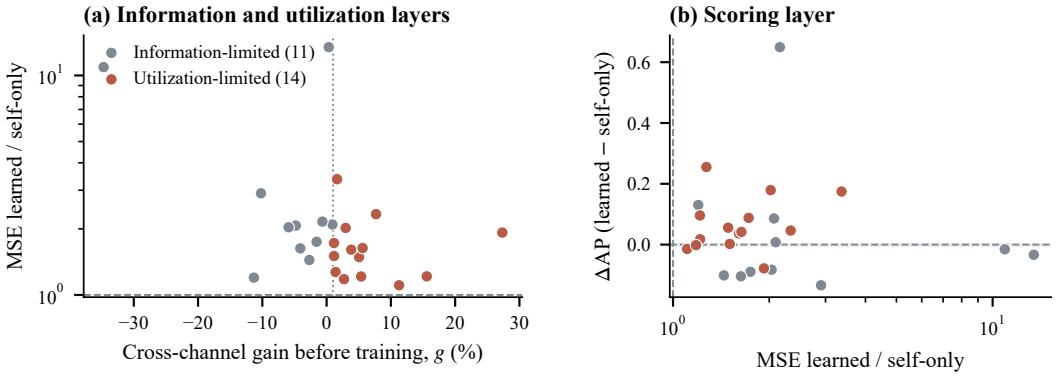


Figure F.1: Layered graph ablation on 25 SMD objects. (a) Pre-training cross-channel gain against the ratio of learned-graph to self-only normal MSE; every object lies above one. (b) The AP difference between learned and self-only support against the same ratio; the gray band marks ties ($|\Delta\mathrm{AP}| \leq 0.005$).

F.2 Where the learned graph's normal error and score come from

This subsection follows the SMD ablation down two further steps: which channels carry the learned graph's normal-error excess over self-only support, and which channels win the maximum in the anomaly score. These two analyses use stored checkpoints and predictions of Appendix F.1; the estimated-graph, capacity, and MTGNN studies below add new fits under protocols committed before their runs.

The channel decomposition of the normal-error gap, score attribution with interpretation labels, and the cost of the diagnosis are reported in Record R13.

Estimated graphs and larger GDN capacity. A protocol committed before the run asks whether GDN's learned graph fails because it selects the wrong sources or because message passing cannot use the right ones. For every target, a pairwise Granger ranking on normal training and selection data (ridge on the target's 16 lags plus one candidate source's 16 lags, gain over the self-history ridge) picks the four best sources; GDN then fits on this fixed support with the standard protocol. Separately, native GDN is refit with top-10 or top-20 neighbors or 64-dimensional embeddings, with a 64-dimensional self-only reference. Table F.2 gives object means over three initializations. The estimated graph lowers normal MSE relative to the learned graph on all 25 objects but stays above self-only support on 23; it beats self-only only on 2-2 and 3-2, two of the three objects with the largest ridge gain (11.3% and 15.6%), and its normal-risk effect tracks g across objects (Spearman 0.56). More neighbors or larger embeddings leave the learned graph above self-only on all 25 objects (median ratios to self-only 1.80, 2.04, and 1.68 for top-10, top-20, and dimension 64; dimension 64 is also above the 64-dimensional self-only model on all 25). AP varies little across these supports. Given a data-estimated graph, GDN's message passing still costs more than the weak cross-channel gain on SMD provides.

A second complete detector: MTGNN. Official MTGNN (pinned commit, unmodified) runs as a one-step forecaster with window 16 on the same 25 objects, data splits, initializations, fitting protocol, calibration, and score, with learned (four sources per target), self-only (zero adjacency), and degree-matched random supports (the GDN random sources); the protocol was committed before the run and development used historical object 1-1 only. The execution record repeats GDN's: with graph learning on, replacing the supplied adjacency changes no prediction in any of the 75 learned fits (maximum difference 0), whereas it changes every self-only and random fit. The null ablation also repeats: AP is 0.414 (learned), 0.410 (self-only), and 0.414 (random); learned minus random is $-0.0001\,[-0.006, +0.006]$ (8/7/10) and learned minus self-only $+0.003\,[-0.006, +0.014]$ (10/5/10). Unlike GDN's, MTGNN's learned graph does not degrade normal prediction: normal MSE is 0.184, 0.185, and 0.184 (median learned/self-only ratio 1.00), half of GDN's learned-graph error and close to GDN's self-only 0.194; self-only normal MSE agrees across the two detectors ob-

Table F.2: **GDN on SMD with estimated and larger supports.** Object means over three initializations; each support has its own normal calibration and the IQR/max score. Counts compare object-mean normal MSE with self-only support of the same embedding size.

Support	Normal MSE	AP	Objects above self-only
Learned, top-5 (Table F.1)	0.368	0.394	25/25
Self-only	0.194	0.346	–
Degree-matched random	0.273	0.399	–
Estimated (pairwise Granger, 4 sources)	0.241	0.376	23/25
Learned, top-10	0.413	0.391	25/25
Learned, top-20	0.475	0.404	25/25
Learned, dimension 64	0.365	0.399	25/25
Self-only, dimension 64	0.190	0.343	–

ject by object (Spearman 0.96). With the same layer rule on the standard budget, MTGNN's labels coincide with GDN's on the 11 information-limited objects and assign graph use to four objects (1-2, 3-8, 2-2, 3-9), all among the seven with ridge gain of at least 5%, where learned-minus-self-only AP is -0.014, $+0.014$, -0.002, and $+0.012$. Across objects, g and MTGNN's normal-risk effect are rank-correlated at 0.30. In MTGNN the replacement is therefore absorbed at the information layer on 11 objects and, where the graph is used, at the scoring layer.

History retention on real data. With normal data only, per-node ridge probes compare self-history with self-history plus the lags of each supplied source, for the executed support of every learned GDN checkpoint, its transpose, and the degree-matched random support. The median gains over self-history are small for all three: 3.8%, 3.2%, and 3.8% at $W = 1$ and 1.9%, 2.1%, and 0.9% at $W = 16$ (learned, transpose, random). Removing the sources' lag 1 at $W = 16$ lowers the learned and transposed gains by similar amounts (0.27 and 0.33 percentage points), and the transpose keeps at least the learned graph's gain on 8 of 25 objects. The transposed learned graph thus retains most of the learned graph's predictive value at every window, as expected from GDN's symmetric selector, which links many pairs in both directions; reversing a learned SMD graph therefore removes little information.

F.3 Information-layer robustness, attention share, and the root-weight repair on SMD

This study answers three questions left by the SMD ablation of Section 6: whether the information-limited labels depend on the tolerance and on the gain estimator, how much of GDN's attention stays on the node itself, and whether a separate root weight closes the gap between the learned graph and self-only support. Its protocol, code, and analysis plan were committed before the formal run. All information estimates, attention readouts, fits, and normal-data verdicts use normal data only and were locked before test scores were computed; the AP comparisons reuse the official SMD test labels opened in earlier studies.

Tolerance and estimator. Relabeling the locked gains at $\tau \in \{0.5, 1, 2, 5\}\%$ gives 10, 11, 15, and 18 information-limited objects, and GDN reaches graph use on no object at any tolerance, because its learned graph raises long-budget normal error on all 25 objects. We then recomputed the gain with four estimators fitted on normal data only (Table F.3): ridge with penalties chosen by blocked cross-validation; a nested ridge, the pre-registered primary alternative, that gives cross-channel lags their own cross-validated penalty and contains the self-history ridge as a special case; ridge on the four normal-data Granger sources; and gradient boosting from the self-history prediction. The pre-registered calls are that the nine negative published gains are an estimation cost if the nested ridge brings at least eight to $g \geq -1\%$ (it brings five, all to $g \geq 0$), and that the information-limited label is robust if at least nine of the eleven objects keep it (seven keep it: partly robust). The remaining four negative gains are a transfer failure between the training and evaluation segments, not input dimension: blocked cross-validation inside the training segment selects finite cross-channel penalties on most of their nodes. Nonlinear estimation finds no additional information (median gain 2.3% for boosting against 2.6% for the nested ridge). Across all estimators and tolerances the information-limited count ranges from 5 to 20, and graph use stays at zero for GDN.

Table F.3: **Information-layer estimators on the 25 SMD objects** (normal data only; GDN labels at $\tau = 1\%$ with the long-budget utilization rule). Kept: of the 11 published information-limited objects, how many keep the label. ρ: Spearman correlation between gain and the learned graph's normal-risk effect, with object-bootstrap 95% interval.

Estimator	$g < 0$	Minimum g	Median g	Information-limited	Kept	ρ
Published ridge	9	-34.7%	1.15%	11	11	$0.35\,[-0.09, 0.69]$
Blocked-CV ridge	12	-44.3%	0.28%	13	11	$0.43\,[-0.02, 0.76]$
Nested ridge (primary)	4	-10.6%	2.59%	7	7	$0.24\,[-0.18, 0.61]$
Granger-4 ridge	4	-9.0%	3.45%	6	6	$0.09\,[-0.34, 0.51]$
Gradient boosting	3	-9.4%	2.27%	7	6	$0.37\,[-0.07, 0.73]$

Attention share of the node itself. All 75 learned and 75 degree-matched random GDN checkpoints are stored, so we read the attention of GDN's single graph layer without retraining, after each checkpoint reproduced its stored normal MSE (largest relative difference 1.6×10^{-7}). With the learned graph the self edge receives 0.212 of the attention on normal data (checkpoint range 0.174–0.236), the uniform value over five inputs, with entropy at 96% of its maximum; with random sources it receives 0.292, and the learned-minus-random difference is $-0.080\,[-0.094, -0.065]$ over objects. The level of dilution is nearly constant across objects, and the pre-registered primary endpoint, a negative correlation across objects between the self share and the learned graph's normal-MSE excess, is not observed (Spearman 0.04, $[-0.37, 0.44]$, one-sided $p = 0.60$). Attention thus shows that self-history is mixed uniformly with the sources, but it does not by itself predict which objects lose most; the intervention below tests the component directly.

Root weight. GDN-root computes $z_i = \text{BN-ReLU}\big(W_{\text{root}} x_i + \sum_{j \in N(i)} \alpha_{ij} W x_j + b\big)$ with the softmax over the four non-self sources only; embeddings, top-k selection, attention logits, output layer, optimizer, budgets, calibration, and score are GDN's, and all shared parameters start from GDN's values at the same seed. Table F.4 reports four supports. The gap to self-only support closes by the pre-registered criterion (median MSE ratio at most 1.01) on every support, the root path carries 54% of the pre-normalization activation, and with the graph's value no longer hidden behind the utilization layer, the information gain predicts the graph's normal-risk effect on every support. AP stays close to that of the original GDN, and a learned graph matches a random one.

Table F.4: **Root-weight GDN on the 25 SMD objects** (three initializations per object). Ratio: median over objects of normal MSE against self-only GDN at the same budget; closure: median fraction of the original support's excess normal MSE removed. Labels at $\tau = 1\%$ (information-limited / utilization-limited / graph used). ρ: Spearman correlation between the published gain and the graph's normal-risk effect. ΔAP against self-only GDN with object-bootstrap 95% interval.

Support	MSE	Ratio	$\leq$ self	Closure	Labels	ρ	AP	ΔAP vs. self
Original learned	0.368	1.72	0	–	11/14/0	$0.35\,[-0.09, 0.69]$	0.394	$+0.048\,[-0.005, +0.116]$
Learned + root	0.194	1.006	9	0.995	11/9/5	$0.48\,[0.03, 0.80]$	0.380	$+0.034\,[+0.0004, +0.069]$
Random + root	0.194	1.004	10	0.993	11/9/5	$0.71\,[0.37, 0.89]$	0.374	$+0.029\,[-0.002, +0.062]$
Learned + root, long	0.190	1.000	13	1.001	11/7/7	$0.61\,[0.21, 0.87]$	0.376	$+0.038\,[+0.007, +0.074]$
Granger-4 + root	0.190	0.991	13	0.994	11/6/8	$0.57\,[0.12, 0.86]$	0.389	$+0.044\,[+0.005, +0.086]$

Paired against the original GDN on the same support, the root weight changes AP by $-0.015\,[-0.057, +0.017]$ (learned), $-0.025\,[-0.077, +0.020]$ (random), and $+0.014\,[-0.023, +0.054]$ (Granger-4); learned against random support with the root weight gives $+0.005\,[-0.009, +0.022]$ (10 wins, 6 ties, 9 losses). Graph use appears on machine-1-6, 2-2, 3-2, 3-7, and 3-9 with the learned support, on the same objects except 1-6 plus 3-8 with random support, and in addition on 1-2, 3-4, and 3-6 with the Granger support. The analysis scripts were written before the formal fits and committed unchanged after the test scores had been computed (Record R7).

All three located components replaced together. The controlled cycle located three components in GDN, and only the root weight had been tried on SMD. Under a protocol committed before any fit, we replaced all three together on the same 25 objects and three initializations: the trainable

directed selector and GATv2 attention of Appendix E.4 and the root weight above, with GDN's SMD configuration and standard budget; a second arm runs GATv2 and the root weight on the degree-matched random support. The analysis script was committed before the test was scored once. Table F.5 gives the results. The learned graph now predicts normal data as well as self-loops (median MSE ratio 0.997) and is used on 9 of 25 objects, against 5 with the root weight alone and none originally. Detection does not follow: learned minus random AP is $-0.014\,[-0.040, +0.012]$, the change against the original GDN is $-0.036\,[-0.094, +0.007]$, and the gain over self-only GDN is $+0.012\,[-0.006, +0.031]$; across objects, the normal-risk gain and the AP gain over self-only correlate at 0.21. On the 9 graph-used objects (a description after the analysis, not a pre-registered endpoint), AP exceeds self-only GDN by $+0.030$ on average (7 of 9 objects higher) and the random graph by $+0.004$. The random support is also used, on 8 objects, so random sources retain the cross-channel gain that the learned graph uses (the redundancy of Section 6), and the gain over self-only reaches AP only weakly.

Table F.5: **GDN on the 25 SMD objects with the three located components replaced together** (trainable directed selector, GATv2 attention, root weight; three initializations per object, standard budget). Top: object-averaged AP and AUROC with the stored original, self-only and root-weight arms. Bottom: paired AP contrasts with 95% object-bootstrap intervals and wins/ties/losses over objects (tie band 0.005). Normal-data labels of the learned arm at $\tau = 1\%$: 11 information-limited, 5 utilization-limited, 9 graph used; median normal MSE relative to self-only 0.997.

Arm	AP	AUROC
Original GDN, learned graph	0.394	0.811
Original GDN, random graph	0.399	0.816
Self-only GDN	0.346	0.821
Root weight, learned graph	0.380	0.828
Root weight, random graph	0.374	0.824
All three replaced, learned (selector)	0.358	0.829
All three replaced, random graph	0.372	0.831
Contrast	**ΔAP**	**W/T/L**
All three: learned − random	$-0.014\,[-0.040, +0.012]$	11/1/13
All three − original (learned)	$-0.036\,[-0.094, +0.007]$	10/1/14
All three (learned) − self-only	$+0.012\,[-0.006, +0.031]$	13/2/10
All three − root weight (learned)	$-0.022\,[-0.050, +0.003]$	8/3/14
All three (random) − self-only	$+0.026\,[-0.003, +0.058]$	17/0/8
Root weight: learned − random	$+0.005\,[-0.009, +0.022]$	10/6/9
Original: learned − random	$-0.005\,[-0.026, +0.015]$	11/1/13

F.4 SWaT AND PEMS-BAY: THE DIAGNOSIS WHERE A GRAPH SHOULD MATTER

This study runs the complete diagnosis on two datasets whose sensors are physically coupled. Its protocol, runner, data manifests, and analysis script were committed before the formal run and before any test score was read.

Data. SWaT (Goh et al., 2017) is the Secure Water Treatment testbed (51 sensors and actuators; seven days of normal operation and four days with 36 attacks), used in the one-minute preprocessed release of GReLeN (Zhang et al., 2022): 5280 normal minutes split in time into training, selection, calibration, and evaluation segments, and a 7499-minute test with 933 attack minutes (prevalence 0.125). GDN's own preprocessing uses 10-second medians, which requires the raw release, so the resolution differs from the GDN paper. Fifteen channels are constant during normal operation and are kept, as in GDN and GReLeN. PEMS-BAY (Li et al., 2018) provides five-minute traffic speeds; we take 64 sensors by breadth-first search on the symmetrized road-network kernel from the best-connected sensor and inject 20 replay attacks with a fixed seed: a sensor's readings over 2–5 hours are replaced by its readings one week earlier, with six-step linear blending at both ends, and an event is kept only when the replayed values differ from the true ones by at least one training standard deviation on average (833 attacked steps in an 8192-step test, prevalence 0.102). The same labels applied to the unattacked test series measure how much AP background traffic anomalies contribute.

The design (detectors, arms, budgets, and the rule set) is specified in Record R13.

Information layer and a change before the formal run. A timing trial on normal data only, with two-epoch fits and before any test was read, showed that the all-channel ridge of the SMD rule predicts SWaT's normal data worse than self-history ($g = -2.2\%$), whereas ridge on only 14 supplied sources gains 21.5%. Because a superset of inputs cannot carry less information, the primary label uses the gain of the supplied sources over self-history, as Algorithm 1 defines it, and the all-channel rule is reported as a secondary label. This change was recorded in the protocol before any PEMS-BAY information estimate, any full fit, or any test score. Supplied-source gains are 21.5% (fixed graph) and 20.5% (random sources) on SWaT and 3.2% and 0.9% on PEMS-BAY; low-dimensional and boosted estimators give 19.3% and 22.9% on SWaT and 2.0% and -5.4% on PEMS-BAY. With the SMD estimators of Appendix F.3, the nested all-channel ridge labels SWaT information-limited, and on PEMS-BAY it agrees with the primary labels.

Table F.6: **Layer labels and AP on SWaT and PEMS-BAY** (three initializations; long budget for GDN pairs, as in the SMD rule). Gain: supplied-source gain over self-history on normal data. Ratio: normal MSE of the graph arm over the self-only arm of the same detector. ΔAP: graph minus self-only, with the two-stage bootstrap interval.

Detector	Graph	Gain	Ratio	Label	AP graph	AP self	ΔAP
SWaT (prevalence 0.125)							
GDN	learned	16.4%	6.95	utilization	0.614	0.570	$+0.044\,[-0.003, +0.152]$
GDN	fixed	21.5%	4.50	utilization	0.478	0.570	$-0.091\,[-0.263, +0.099]$
GDN + root	learned	14.9%	1.24	utilization	0.415	0.718	$-0.303\,[-0.508, -0.004]$
GDN + root	fixed	21.5%	1.19	utilization	0.536	0.718	$-0.182\,[-0.431, +0.052]$
MTGNN	learned	8.6%	1.59	utilization	0.594	0.345	$+0.250\,[+0.015, +0.380]$
MTGNN	fixed	21.5%	1.51	utilization	0.514	0.345	$+0.169\,[-0.009, +0.336]$
PEMS-BAY with replay attacks (prevalence 0.102)							
GDN	learned	1.3%	4.88	utilization	0.151	0.123	$+0.028\,[-0.015, +0.069]$
GDN	road	3.2%	3.32	utilization	0.163	0.123	$+0.040\,[-0.001, +0.116]$
GDN + root	learned	1.1%	0.99	used; scoring	0.123	0.124	$-0.001\,[-0.005, +0.003]$
GDN + root	road	3.2%	0.98	used; scoring	0.123	0.124	$-0.000\,[-0.003, +0.002]$
MTGNN	learned	2.7%	0.98	used; scoring	0.118	0.123	$-0.005\,[-0.015, +0.001]$
MTGNN	road	3.2%	0.98	used; scoring	0.122	0.123	$-0.001\,[-0.006, +0.004]$

Results. Table F.6 gives the labels. In every learned-graph arm of GDN and MTGNN, swapping the supplied graph leaves the executed edges unchanged (largest prediction difference 4.3×10^{-6}), whereas MTGNN with graph learning off executes it. The random-graph refit changes GDN's AP by $-0.070\,[-0.232, +0.125]$ on SWaT and $+0.005\,[-0.027, +0.037]$ on PEMS-BAY, and MTGNN's by $+0.017\,[-0.067, +0.067]$ and $-0.000\,[-0.004, +0.004]$. On PEMS-BAY every detector's AP is close to prevalence and matches its AP on the unattacked series with the same labels (differences within 0.013), so the forecasting residual responds to background traffic anomalies rather than to the replays; the repaired GDN and MTGNN use the graph for prediction, and the replacement is absorbed at scoring. On SWaT the root weight reduces GDN's excess normal error from 6.95 to 1.24 times self-only but does not remove it, and the graph lowers the repaired GDN's AP; MTGNN's learned graph raises AP by 0.25 without lowering normal risk, and random sources do the same. Every arm, initialization, and failed attempt is retained: the local SWaT fit stopped once on a transient CUDA error after 25 of 33 fits, and a resume driver that reuses the frozen fit code completed the remaining eight, keeping the interrupted attempt; seed-level results are in the result files.

Ten initializations, an official no-graph control, and channel-mean scoring. A follow-up registered before any of its fits (protocol, runner, and analysis committed first) fits MTGNN with seven further initializations (713–719) for the four supports, adds the official no-graph MTGNN (graph convolutions replaced by the official residual 1×1 convolutions; 162,017 parameters against 180,641) for all ten initializations, and scores every fit both with the channel maximum used above and with the channel mean of the same standardized errors. The three earlier fits per support are reused unchanged. Over ten initializations, learned minus random support is $+0.020$, and the learned graph's AP exceeds self-loops by $+0.095$ (t interval $[-0.091, +0.281]$;

5 wins, 2 ties, 3 losses): the $+0.25$ of the first three initializations becomes $+0.029$ on the seven new ones. The zero-adjacency control is not degenerate relative to official no-graph MTGNN, whose AP is lower ($-0.105\,[-0.257, +0.048]$), and every graph support exceeds the no-graph model (learned: $+0.200\,[+0.029, +0.370]$). With channel-mean scoring every MTGNN variant reaches AP 0.707–0.712, above every channel-max variant (0.329–0.544), and the graph effect is $+0.004\,[-0.003, +0.010]$. Channel-max AP varies across initializations with a standard deviation of about 0.19 for every graph support. By the pre-registered reading, the SWaT difference is located at the scoring layer, in the channel-maximum aggregation, and does not establish a graph effect on detection; the three-initialization result above is retained as reported.

Scoring transfer to SKAB and PSM. A study that locks the SMD score recipe and transfers it to SKAB and PSM, where the 5% rule changes AP by -0.3101 and $+0.0639$ and the PSM gain depends on predictor-specific calibration, is retained in full, with all failures, in Record R4.2.

G Extended Related Work and Discussion

This section expands the relation to prior work and states the conditions behind each main-text result with their numbers.

G.1 Extended discussion and limitations

Mechanism scope. Propositions 1 and 2 are direct consequences of the permutation lag covariance $\Gamma_k = \rho^k P^k$, and Proposition 3 is the standard conditional-projection identity; their role is to make each diagnostic layer measurable, not to extend VAR theory. Retention is one information-layer mechanism, isolated in the controlled cycles; the SMD null result arises at other layers (Section 6). The first-lag result is a property of the cycle dynamics used to build the controlled testbed. Multiple paths alter the lag, and common ancestors or path cancellation can create or remove predictive gain (Records R9.1 and R9.2). In random sparse VARs, a reversed graph retains little detection: AP is at most 0.53 at $W = 4$, against at least 0.99 for the true graph (Appendix B.3). The residual spectral analysis applies to zero-mean Gaussian residuals and squared scores.

Complete detectors. We attribute lag use in GDN with fixed oracle sources (five data seeds, one initialization) and with native learned graphs (five further data seeds, three initializations); MTGNN is the second complete detector, on SMD and on the controlled cycle. Native child effects are small: the frozen all-node child contrast is 2.12×10^{-4}, and the refitted child contrast is $0.00059\,[-0.00237, 0.00368]$, an interval that crosses zero (Record R10.1). The twenty-seed confirmation fixes one initialization; it confirms the sign of both frozen contrasts in every seed while full-model AP stays at 0.515 (Record R10.2); the rerun of native GDN in Appendix E.4, on the same packets, reaches 0.545, and the difference reflects non-deterministic GPU training of a selector without direct gradient rather than a change in setup. Native GDN exposes the parent to about as many targets as four randomly drawn sources would (4.06). The earlier MTGNN comparison changed the graph constructor, propagation, and temporal model together (Record R9.4); the single-component replacements of Appendix E.4 isolate the selector, and a trainable selector alone raises AP to 0.95. Source identities come from the controlled dynamics, and identifying the corresponding relations in a real system requires additional information and matched interventions.

Real data. On SMD, the information layer is a linear ridge estimate on 608 inputs; its gain is negative on 9 objects, so an information-limited label marks weak linearly estimable gain at this sample size. The primary rank correlation between this gain and the graph's normal-risk effect is 0.35, with a bootstrap interval $[-0.09, 0.69]$ that crosses zero, and the self-only AP difference, $+0.048\,[-0.005, +0.116]$, also crosses zero (Appendix F.1); both are reported descriptively. SMD test labels had already been opened in a preliminary study of predictor adoption (Record R9.5), so the graph ablation reuses the official tests rather than providing an independent test. The SMD objects come from one dataset, and its labels do not supply relational-change ground truth; the history and representation adjustments that lower normal risk on real objects do not identify the controlled four-cycle mechanism as the cause of real anomalies. SKAB records share one physical testbed and PSM contributes one complete test, each with three initializations, and their score recipe was selected on three historical SMD objects with their labels (Record R4.2).

Scoring and thresholds. Lower normal risk, ranking gains, and threshold transfer are separate outcomes. On the 25-object SMD cohort, nominal 1% normal-calibration exceedance gives test FPRs of 30.38% for the original GDN and 13.81% for the 5% rule (Record R9.5). On SKAB the adopted rules keep FPR at 90.02%–98.78%, and on PSM the comparator's recall at its locked threshold is 0.50% (Record R4.2). Risk intervals assume approximately stationary, weakly dependent losses; in short, strongly correlated Gaussian simulations, nominal 99% intervals for fixed predictors covered 91%–92% (Record R2.5), so we use them for screening and evaluate gains on held-out data.

H Reproducibility and Supplementary Records

Every experiment keeps per-seed outputs, configurations, input manifests, and checkpoints or decision receipts, organized as in Table R8.5, and scripts regenerate every reported metric from saved predictions. Protocols and code were committed before the formal runs of the SMD graph ablation, the controlled MTGNN test, the second-detector checks on SMD, the single-component replacements with their two addenda, the SMD robustness and root-weight study, the SWaT and PEMS-BAY study, the GraGOD execution study, the importance readouts on the planted benchmark, and the three-component replacement on SMD. Analysis scripts were committed before the test scores they read, except in the root-weight study, whose scripts were written before its tests were scored and committed unchanged afterwards (Appendix F.3). The supplementary code also contains a detector-agnostic implementation of Algorithm 1 with tests.

The supplementary records, a separate PDF, keep every study and development record that this appendix omits, without deletion. Their items are numbered with an R (for example, Table R4.13), and their first page gives a reading guide.

References

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv preprint arXiv:1803.01271*, 2018. URL https://arxiv.org/abs/1803.01271.

Federico Bello, Gonzalo Chiarlone, Marcelo Fiori, Gastón García González, and Federico Larroca. GNNs for time series anomaly detection: An open-source framework and a critical evaluation. In *Proceedings of the 15th International Conference on Pattern Recognition Applications and Methods*, pp. 53–61. SCITEPRESS, 2026. doi: 10.5220/0014312000004067. URL https://www.scitepress.org/PublishedPapers/2026/143120/.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

Zekai Chen, Dingshuo Chen, Xiao Zhang, Zixuan Yuan, and Xiuzhen Cheng. Learning graph structures with Transformer for multivariate time-series anomaly detection in IoT. *IEEE Internet of Things Journal*, 9(12):9179–9189, 2022. doi: 10.1109/JIOT.2021.3100509.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

Enyan Dai and Jie Chen. Graph-augmented normalizing flows for anomaly detection of multiple time series. In *International Conference on Learning Representations*, 2022.

Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4027–4035, 2021. doi: 10.1609/aaai.v35i5.16523. URL https://ojs.aaai.org/index.php/AAAI/article/view/16523.

Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. A dataset to support research in the design of secure water treatment systems. In *Critical Information Infrastructures Security (CRITIS 2016)*, volume 10242 of *Lecture Notes in Computer Science*, pp. 88–99. Springer, 2017. doi: 10.1007/978-3-319-71368-7_8.

Siho Han and Simon S. Woo. Learning sparse latent graph representations for anomaly detection in multivariate time series. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2977–2986, 2022. doi: 10.1145/3534678.3539117.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJiHXGWAZ.

Pablo Morales-Mombiela, Daniel Hernández-Lobato, and Alberto Suárez. Statistical tests for the detection of the arrow of time in vector autoregressive models. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pp. 1544–1550, 2013. URL https://www.ijcai.org/Proceedings/13/Papers/230.pdf.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for NLP with differentiable edge masking. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=WznmQa42ZAx.

Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9:307, 2008. doi: 10.1186/1471-2105-9-307.

Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2828–2837, 2019. doi: 10.1145/3292500.3330672. URL https://doi.org/10.1145/3292500.3330672.

Irene Winkler, Danny Panknin, Daniel Bartz, Klaus-Robert Müller, and Stefan Haufe. Validity of time reversal for testing Granger causality. *IEEE Transactions on Signal Processing*, 64 (11):2746–2760, 2016. doi: 10.1109/TSP.2016.2531628. URL https://arxiv.org/abs/1509.07636.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 753–763, 2020. doi: 10.1145/3394486.3403118. URL https://doi.org/10.1145/3394486.3403118.

Weiqi Zhang, Chen Zhang, and Fugee Tsung. GReLeN: Multivariate time series anomaly detection from the perspective of graph relational learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 2390–2397, 2022. doi: 10.24963/ijcai.2022/332. URL https://www.ijcai.org/proceedings/2022/332.