AI and Data Science Journal
ISSN: 3078-7394
DOI: https://doi.org/10.61784/adsj4038

DAS-GNN: A SCALABLE DISAGREEMENT-AWARE GRAPH NEURAL FRAMEWORK FOR ANOMALY DETECTION IN LARGE-SCALE NETWORKS

BoYuan Wang[1*], WenYu Zhao[2], ZiMeng Wang[3]
[1]*University of Southern California, Los Angeles 90089, CA, USA.*
[2]*Microsoft, Beijing 100080, China.*
[3]*Brandeis University, Waltham 02453, Massachusetts, United States.*
[*]*Corresponding Author: BoYuan Wang*

Abstract: Graph neural networks can exploit relational context for anomaly detection, but repeated neighborhood expansion and dense graph construction make many designs difficult to deploy on large network-flow collections. This paper presents DAS-GNN, a decoupled disagreement-aware framework that converts flow records into a deterministic, degree-capped locality graph and performs graph propagation only once. The graph is built without labels by combining categorical signatures with multiple random-projection orderings, avoiding an all-pairs nearest-neighbor search. A cosine-gated context channel estimates locally stable behavior, while an explicit deviation channel preserves the absolute disagreement between each record and its neighborhood together with similarity, entropy, and categorical-agreement statistics. The resulting graph views are trained with ordinary dense mini-batches, so each epoch is independent of edge traversal. Experiments use the official UNSW-NB15 partition with 257,673 labeled network flows. Across three seeds, DAS-GNN obtains 0.9721±0.0004 ROC-AUC, 0.9794±0.0003 PR-AUC, and 0.8771±0.0022 F1, improving the matched MLP by 0.0042, 0.0029, and 0.0083, respectively. A strong LightGBM baseline remains best at 0.9852 ROC-AUC, which bounds the claim: the contribution is scalable graph-context enhancement rather than universal dominance over tabular boosting. On the complete graph, construction plus propagation takes 1.59 s in a controlled CPU scaling run; a 500,000-node stress graph takes 4.46 s.
Keywords: Anomaly detection; Graph neural networks; Network intrusion detection; Scalable graph learning; Decoupled propagation

1 INTRODUCTION

Anomaly detection in communication and transaction networks must operate under three simultaneous pressures: a rapidly growing number of events, highly imbalanced or shifting behaviors, and the need to preserve relational evidence that is absent from an isolated feature vector. Classical detectors such as local density estimation and isolation-based ensembles remain useful, while deep anomaly detection provides flexible nonlinear representations [1-3]. However, a flow table is not automatically a graph, and a graph model is not automatically scalable. A practical framework must therefore answer two engineering questions before adding architectural depth: how can a sparse relation graph be constructed without quadratic search, and how can neighborhood information be reused without traversing the graph in every training epoch?

Message-passing GNNs provide a natural language for relational context. GCN, GraphSAGE, and graph attention aggregate neighboring features to create task-specific representations [4-6]. Yet multi-layer expansion can repeatedly fetch the same neighbors and can make memory grow with fan-out. Sampling methods such as FastGCN, LADIES, Cluster-GCN, and GraphSAINT reduce this cost through layer or subgraph sampling [7-10]. Precomputation-oriented models, including SGC, SIGN, and PPRGo, instead move propagation outside the optimizer loop [11-13]. These ideas motivate a detector in which graph structure is a bounded preprocessing resource rather than an epoch-dependent computational obligation.

Graph anomaly detection introduces another complication. Anomalies often disagree with local context, but naive averaging may erase precisely the feature residual that signals abnormality. Reconstruction methods such as DOMINANT and AnomalyDAE model attribute and structural errors; contrastive and integrity-oriented methods such as CoLA, CARE-GNN, and PC-GNN learn context or relation-aware recognition [14-18]. GUIDE further models higher-order structure [19]. These approaches demonstrate the value of relational inconsistency, although most assume a native attributed graph. In network-flow benchmarks, by contrast, the published data may be a table. A graph must be induced, and the induction procedure becomes part of the scientific claim.

This paper proposes DAS-GNN (Disagreement-Aware Scalable GNN), a fixed-degree, decoupled framework for labeled anomaly detection on large flow collections. It uses protocol, service, and state to define coarse behavioral strata. Within each stratum, multiple random projections generate short candidate lists by local ordering; the final K neighbors are selected in a compact high-variance numeric subspace. This procedure avoids materializing all pairwise distances. Next, DAS-GNN computes a similarity-gated neighborhood mean, an elementwise deviation vector, and compact context statistics once. A small neural detector then operates on precomputed features in ordinary mini-batches.

The framework is evaluated on the official UNSW-NB15 training and testing CSV partitions [20]. The dataset contains normal traffic and nine attack categories, and its creators describe the corpus as a hybrid of real normal activity and synthetic contemporary attack behavior. We therefore avoid characterizing all labels as naturally occurring production incidents. The main experiment is transductive with respect to unlabeled features: graph construction can observe feature vectors from the full official partition, but labels and attack-category names are excluded. Preprocessing statistics and all learned parameters are fitted using the official training partition and its internal validation split.

The contributions are fourfold. First, the paper specifies a reproducible multi-view graph induction algorithm whose candidate count and stored degree are constants. Second, it introduces paired stable-context and disagreement channels; the latter prevents aggregation from suppressing anomalous residuals. Third, it decouples propagation from optimization, making per-epoch training independent of the number of stored edges. Fourth, it reports accuracy, ablation, edge-corruption, wall-clock, and memory results without hiding a stronger non-graph baseline. The empirical conclusion is deliberately bounded: graph context consistently improves the matched MLP, but LightGBM remains the most accurate model on this particular tabular benchmark (Figure 1) [21].

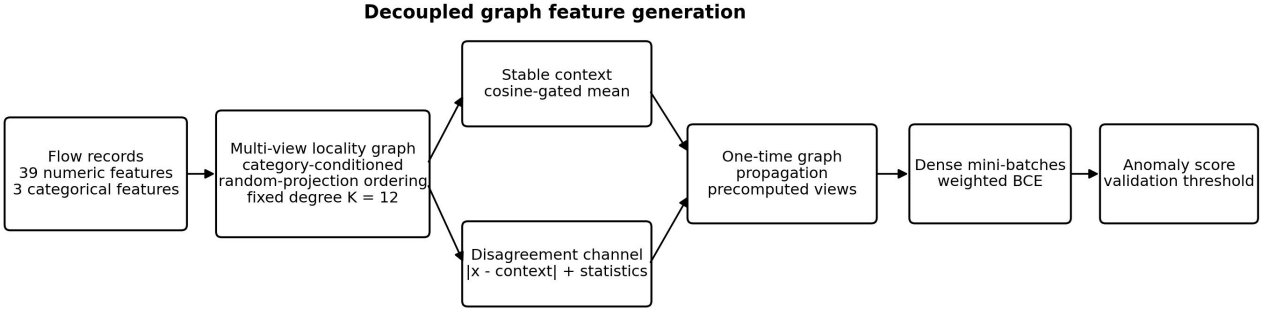

Figure 1 DAS-GNN Separates Label-Free Graph Construction and One-Time Propagation from Mini-Batch Detector Training

2 RELATED WORK

2.1 Anomaly Detection and Graph Anomalies

Graph-based anomaly detection spans unusual nodes, edges, subgraphs, and whole graphs. Akoglu et al. organized early methods around structural and attribute-based deviations, while later surveys describe the transition toward deep graph models and identify scalability, contamination, and evaluation as persistent challenges [1, 22-23]. Public resources such as PyGOD improve implementation access, and large financial datasets such as DGraph reveal the importance of scale, temporal structure, and partially labeled nodes [24-25]. These developments support the use of graph context, but they also warn against comparing results across incompatible anomaly definitions or graph-construction assumptions.

Deep attributed-network detectors frequently optimize reconstruction or self-supervision. DOMINANT jointly reconstructs topology and attributes, and AnomalyDAE uses dual autoencoders [14-15]. CoLA constructs contrastive local views, whereas GUIDE includes higher-order structural patterns [16, 19]. Risk prevention has emphasized imbalance and relation noise: CARE-GNN selects informative relations, and PC-GNN combines class-balanced sampling with neighborhood aggregation [17-18]. DAS-GNN differs in scope. It is a supervised flow-level detector for data that do not provide a native adjacency matrix; consequently, its graph is label-free but constructed from observed flow attributes. Direct numerical comparisons with unsupervised native-graph methods would not be controlled, so the experiments instead use tabular and graph ablations sharing exactly the same split and neural backbone.

2.2 Scalable Graph Learning

Neighbor sampling in GraphSAGE bounds fan-out, while FastGCN and LADIES sample nodes according to importance [5, 7-8]. Cluster-GCN forms dense subgraph mini-batches, and GraphSAINT samples training subgraphs with normalization [9-10]. PinSAGE demonstrates random-walk-based neighborhood selection at web scale [26]. A complementary line removes message passing from the optimizer loop: SGC collapses linear propagation, SIGN precomputes multiple graph operators, and PPRGo uses sparse approximate personalized PageRank [11-13]. DAS-GNN follows the second philosophy. Its graph views are computed once on CPU, after which training is equivalent to a compact MLP over dense arrays. The distinction is important for anomaly detection because multiple validation thresholds, seeds, or cost functions can reuse the same graph cache.

2.3 Evaluation Under Class Imbalance

ROC-AUC is threshold independent but can look optimistic when negatives dominate; precision-recall analysis emphasizes performance on the positive class [27-28]. UNSW-NB15 is not an extreme rare-event split—the attack class is the majority in the official training set—yet normal and attack costs remain asymmetric. We report ROC-AUC, average precision (PR-AUC), F1, precision, and recall. Training uses symmetric inverse-frequency weights rather than

treating either class as cost-free, following general recommendations for imbalanced learning [29]. Thresholds are selected only on validation data.

3 PROBLEM FORMULATION AND DESIGN GOALS

Let X be the N-by-d numeric flow matrix, C the categorical attributes, and y(i) in {0,1} the normal/attack label available for training nodes. The input does not include a native edge set. We construct a directed fixed-degree graph G=(V,E), with |V|=N, in which every node stores K outgoing neighbors. The detector estimates the probability that y(i)=1 from x(i), c(i), and G. Attack-category strings are never used as features. The official testing labels remain hidden during preprocessing, model selection, and threshold selection.

Four design goals guide the framework. (G1) Graph induction must not require $O(N^2)$ pairwise distances. (G2) memory should be predictable from N, K, and d. (G3) graph aggregation must expose both agreement and disagreement because an anomalous flow may be close to benign neighbors in some dimensions while sharply deviating in others. (G4) the learned detector should train with conventional mini-batches so that repeated experiments do not repeatedly expand neighborhoods. These goals favor a shallow, explicitly decoupled model over a deeper end-to-end message-passing stack.

4 DAS-GNN FRAMEWORK

4.1 Training-Only Feature Transformation

The 39 numeric columns are cleaned by replacing non-finite values and filling missing values with zero. For a nonnegative heavy-tailed feature whose 99th percentile is more than twenty times its median (with a small numerical guard), log(1+x) compression is applied. A robust scaler with the 5th–95th percentile range is fitted on the official training records, and transformed values are clipped to [−8,8]. The three categorical variables—protocol, service, and state—are integer encoded from training categories with a reserved unknown token. This transformation is shared by all neural variants.

4.2 Degree-Capped Multi-View Locality Graph

An exact k-nearest-neighbor graph over all numeric dimensions would require expensive indexing or large candidate searches. DAS-GNN instead constructs a small deterministic candidate set. First, it selects d_s=min(16,d) numeric dimensions with the largest variance. Second, the categorical codes are combined into a mixed-radix signature q_i, creating coarse groups of flows that share protocol, service, and state. For each of R=4 fixed Gaussian projection vectors r_v, a scalar projection z_i^{(v)}=x_{i,S}^T r_v is computed. Nodes are lexicographically ordered by (q_i,z_i^{(v)}). Up to three preceding and three following positions in each view become candidates when their category signature matches. Immediate acquisition-order neighbors are added as two additional candidates; this weak temporal-locality heuristic uses no label.

The candidate pool contains at most 2RW+2=26 entries for R=4 and W=3. For each node, squared distance is evaluated only in the selected numeric subspace and the K=12 closest candidates are retained. Duplicate or boundary candidates can occur, but the stored adjacency always has exactly K integer entries, which makes memory allocation predictable. The procedure is label-free and avoids an all-pairs matrix. It is transductive because the official test feature vectors participate in ordering; Section 7 discusses this limitation and an inductive deployment variant.

$$N(i) = \text{TopK over } j \text{ in } Q(i) \text{ of } \|x(i,S) - x(j,S)\|^2 \quad (1)$$

4.3 Stable Context by Similarity Gating

A uniform neighbor mean can be contaminated by an incorrect edge. DAS-GNN therefore computes cosine similarity between a node and each neighbor after numeric preprocessing. With temperature $\tau=0.5$, similarities are normalized into a local softmax. The stable context is the weighted mean of neighboring numeric features. Because the gate is computed without labels or trainable parameters, it can be cached together with the adjacency and reused across training runs.

$$a(i,j) = \exp(\cos(x(i),x(j))/\tau) \,/\, \Sigma[k \text{ in } N(i)] \exp(\cos(x(i),x(k))/\tau) \quad (2)$$

$$g(i) = \Sigma[j \text{ in } N(i)]\ a(i,j)x(j) \quad (3)$$

4.4 Disagreement-Aware Channel

The central modeling choice is to preserve the residual that standard smoothing tends to remove. DAS-GNN stores the elementwise absolute deviation d_i=|x_i−g_i|. It also computes seven compact context statistics: mean absolute neighbor difference, mean and standard deviation of cosine similarity, normalized gate entropy, and one agreement ratio for each of the three categorical attributes. High entropy indicates no single neighbor dominates; low categorical agreement can expose a locality graph that is numerically close but behaviorally heterogeneous. The stable context and deviation describe complementary hypotheses: g_i estimates expected local behavior, whereas d_i records how the observation violates that expectation.

$$d(i)=|x(i)-g(i)|; \quad s(i)=[\text{mean difference, cosine mean/std., entropy, three category agreements}] \quad (4)$$

4.5 Decoupled Detector and Objective

The final dense input is u_i=[x_i,g_i,d_i,s_i]. Learned embeddings of protocol, service, and state are concatenated with u_i. A two-hidden-layer network with LayerNorm, ReLU, dropout, and a scalar logit produces the attack probability. The model is intentionally compact: graph reasoning has already been materialized in u_i, so mini-batches do not gather neighbors. AdamW optimization minimizes class-balanced binary cross entropy [30]. If n_0 and n_1 are the class counts in the training subset, the weight of class c is n/(2n_c). Validation PR-AUC determines early stopping, and the F1-maximizing validation quantile determines the final threshold.

$$p(i) = \text{sigmoid}(f\_\text{theta}([x(i), g(i), d(i), s(i), \text{category embeddings}])) \quad (5)$$

$$L = -\Sigma[i \text{ in train}]\ w(y(i))\{y(i)\log p(i) + [1-y(i)]\log[1-p(i)]\} \quad (6)$$

Algorithm 1 Degree-Capped DAS-GNN Precomputation

Step	Operation
1	Fit numeric transforms and category maps on training records.
2	Choose up to 16 high-variance numeric dimensions; form mixed-radix category signature.
3	For each of four fixed random projections, lexicographically order nodes by signature and projection.
4	Collect matching-signature candidates within offsets ±1, ±2, ±3; add immediate sequence candidates.
5	Retain K=12 candidates with minimum compact-subspace squared distance.
6	In chunks, compute cosine weights, stable context, absolute deviation, and seven context statistics.
7	Cache [x, context, deviation, statistics] and train dense mini-batches without graph traversal.

4.6 Complexity

With fixed R, W, K, and d_s, projection and candidate evaluation are linear in N; the dominant theoretical term is sorting R orderings, O(RN log N). Candidate distance selection costs O(N(2RW+2)d_s), and one-time propagation costs O(NKd). The adjacency cache requires O(NK) integers, while graph views require O(Nd) floating-point values. Most importantly, detector training contains no edge traversal: each epoch costs O(Nh(d'+h)) for dense input width d' and hidden width h. This decoupling is advantageous when the same graph supports multiple seeds, thresholds, or ablations.

4.7 Deployment Modes and Cache Refresh

The evaluated mode is batch transductive scoring: a bounded collection of unlabeled flows is available, graph views are generated once, and one or more detectors score the batch. This mode suits periodic security analytics and offline model comparison. A strict inductive service can retain the training-side projection orderings as searchable arrays. An arriving record is encoded with the training maps, inserted virtually into each ordering by binary search, and connected only to nearby training or previously accepted reference nodes. The detector input has the same width, so the learned dense network does not change. The cost shifts from one global sort to R logarithmic searches plus a constant candidate comparison per event.

For a drifting stream, a cache can be refreshed by time window rather than rebuilt after every event. Because K is fixed, replacing a window of M nodes changes O(MK) adjacency entries and O(Md) graph-view values. The current implementation performs a full deterministic rebuild because it is simpler to audit and takes seconds at the evaluated scale. Incremental maintenance is an engineering extension, not an experimentally validated result in this paper. The separation between graph cache and detector nevertheless makes the update boundary explicit: a graph refresh can be scheduled independently from a model retraining cycle.

The cached disagreement vector also offers a limited explanation path. For a high-scoring flow, an analyst can rank numeric dimensions by |x_i−g_i| and inspect categorical agreement and gate entropy. These quantities are not causal explanations, and a downstream neural head can combine them nonlinearly, but they identify which local expectations were violated. This is more operationally interpretable than exposing only a latent neighbor embedding and requires no extra explainer model.

5 EXPERIMENTAL METHODOLOGY

5.1 Dataset and Split

UNSW-NB15 provides 175,341 official training flows and 82,332 official testing flows, for 257,673 nodes after concatenation [20]. The labels comprise normal traffic and nine attack categories: Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms. The training file contains 119,341 attack and 56,000 normal records; the testing file contains 45,332 attack and 37,000 normal records. Fifteen percent of the official training file is reserved by stratified sampling for validation, leaving 149,039 training nodes and 26,302 validation nodes. The test set is never used for optimization or threshold selection (Table 1).

Table 1 Data and Implementation Configuration

Item	Setting
Official train / test	175,341 / 82,332 flows
Numeric / categorical	39 / 3 features
Train / validation	149,039 / 26,302 nodes
Graph	K=12; 3,092,076 directed entries
Views / ordering window	4 / ±3 positions
Neural seeds	7, 21, 42
Optimizer	AdamW; lr=10^{-3}; wd=10^{-4}
Batch / maximum epochs	4096 / 12
Hardware	5 vCPU AMD EPYC 9V74; CPU only

5.2 Baselines and Ablations

Logistic regression uses standardized numeric variables, one-hot categorical variables, balanced class weights, and the same 15% validation protocol. LightGBM is included because tree boosting is a strong fit for heterogeneous tabular traffic features [21]. The neural MLP shares the proposed model's numeric preprocessing, categorical embeddings, optimizer, batch size, and hidden capacity but receives no graph views. PreMean-GNN concatenates x_i with an unweighted neighbor mean. The no-gate ablation replaces the cosine gate with a uniform mean while retaining deviation statistics. The no-deviation ablation retains x_i and the gated context but removes the explicit residual and seven statistics. This design isolates the contribution of graph smoothing, gating, and disagreement without confounding changes in data split.

Neural results are reported as population mean±standard deviation across seeds 7, 21, and 42. Logistic regression and LightGBM are single seed-7 reference runs and therefore have no uncertainty estimate. This asymmetry is reported rather than disguised; the tree result should be interpreted as a strong point baseline, not a statistically ranked winner. The implementation uses Python 3.13, NumPy 2.3.5, pandas 2.2.3, scikit-learn 1.8.0, PyTorch 2.10.0 (CPU), and LightGBM 4.6.0.

5.3 Metrics and Stress Tests

ROC-AUC and PR-AUC evaluate ranking. F1, precision, and recall use a threshold chosen on validation data from 193 score quantiles between 0.02 and 0.98. The robustness test randomly replaces 0%, 10%, 20%, or 40% of neighbor-array entries and retrains PreMean-GNN and DAS-GNN with seed 7; it is a single-seed stress test rather than a confidence interval. Scalability is measured at 25k, 50k, 100k, 257,673, and 500k nodes. Sizes up to the complete dataset are real subsamples. The 500k case bootstraps observed feature rows and adds Gaussian jitter with standard deviation 0.01 only to measure preprocessing scale; no detection accuracy is reported on synthetic stress data.

5.4 Leakage Controls and Reproduction Protocol

Four controls prevent label leakage. First, the columns id, attack_cat, and label are removed before model input. Second, numeric scaling is fitted only on the official training file. Third, validation nodes are sampled only from the official training file; the official test labels are not consulted until final metric calculation. Fourth, graph construction uses only transformed numeric values and the three ordinary categorical fields. The transductive access to test features is disclosed because it is a modeling assumption, but it does not expose test labels or attack names. All methods use the same official test rows, and all neural ablations reuse the same graph and split.

A clean reproduction begins by verifying the two raw-file checksums recorded in the package, creating the pinned Python environment, and running the tabular reference models once with seed 7. Neural variants are then executed separately for seeds 7, 21, and 42 so that a failed long process cannot silently discard completed runs. The merger script concatenates machine-readable rows and computes population standard deviation with no hard-coded metric values. Robustness and scaling commands write separate CSV files. Finally, the plotting script reads only these archived CSVs. This workflow makes every table entry traceable to either a run row or a measured scaling row.

6 RESULTS AND DISCUSSION

6.1 Detection Performance

Table 2 and Figure 2 summarize the official test results. LightGBM achieves the highest scores: 0.9852 ROC-AUC, 0.9891 PR-AUC, and 0.8968 F1. This result confirms that the dataset contains strong nonlinear feature thresholds and interactions that tree boosting can exploit. DAS-GNN is not presented as a universal replacement for boosting. Instead, the controlled neural comparison answers whether bounded graph context adds information beyond the same MLP backbone.

Table 2 Official UNSW-NB15 Test Performance

Method	ROC-AUC	PR-AUC	F1
Logistic regression	0.9518	0.9641	0.8512
LightGBM	0.9852	0.9891	0.8968
MLP	0.9679±.0009	0.9765±.0006	0.8688±.0029
PreMean-GNN	0.9706±.0010	0.9785±.0007	0.8667±.0015
DAS, no gate	0.9721±.0005	0.9794±.0004	0.8786±.0026
DAS, no deviation	0.9705±.0008	0.9784±.0006	0.8648±.0014
DAS-GNN	0.9721±.0004	0.9794±.0003	0.8771±.0022

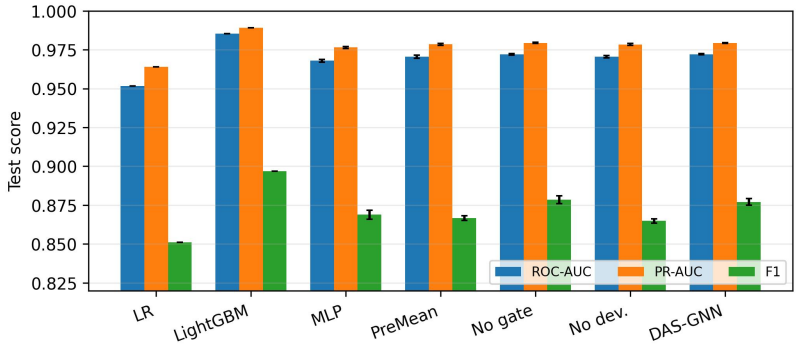

Figure 2 Ranking and Thresholded Test Metrics. Neural error bars show three-seed population standard deviation; tabular baselines are single runs.

Relative to the MLP, DAS-GNN improves ROC-AUC by 0.00419, PR-AUC by 0.00293, and F1 by 0.00827 on average. The ROC-AUC gain is positive for every matched seed: +0.00321, +0.00457, and +0.00478. The corresponding F1 gains are +0.00506, +0.01092, and +0.00884. Although modest in absolute scale, the consistency and low variance indicate that the graph channels are not benefiting from one favorable initialization. Test inference for DAS-GNN averages 0.0269 s after graph views are cached, versus 0.0173 s for the MLP.

PreMean-GNN raises ranking quality over the MLP but slightly lowers F1. Its recall reaches 0.9892, while precision falls to 0.7712. Thus, uniform smoothing makes attack scores easier to rank globally but produces a less favorable validation-selected operating point. DAS-GNN increases precision to 0.7930 while retaining 0.9812 recall, explaining the F1 recovery. In deployment, the threshold should be selected from operational false-alarm costs rather than F1 alone; the reported threshold is an experiment control.

6.2 Ablation Analysis

Removing the deviation channel reduces ROC-AUC from 0.9721 to 0.9705 and F1 from 0.8771 to 0.8648. This is the largest ablation loss and supports the hypothesis that the residual between a flow and its neighborhood contains discriminative evidence that smoothing alone suppresses. Removing the cosine gate has almost no ranking penalty: ROC-AUC changes from 0.972101 to 0.972083, and PR-AUC from 0.979414 to 0.979364. The uniform-context variant even has a 0.00146 higher mean F1. Consequently, the gate should be interpreted as a robustness and ranking component, not as the sole source of accuracy. A future learned gate may improve threshold calibration, but the current label-free gate is intentionally cheap and reproducible.

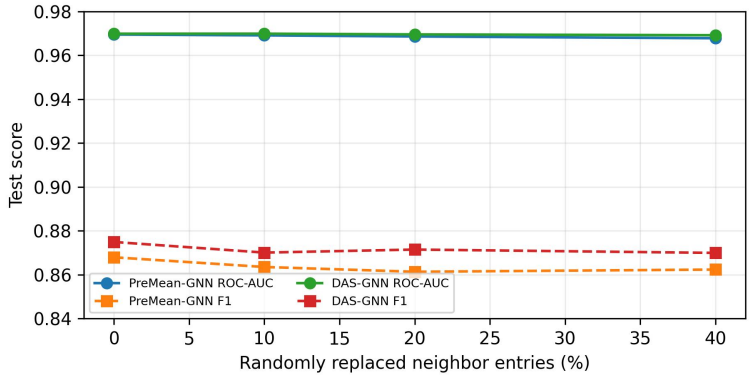

Figure 3 Single-Seed Graph-Corruption Stress Test. Neighbor entries are replaced uniformly at random before graph-view precomputation.

6.3 Robustness to Edge Corruption

Figure 3 perturbs the graph rather than the input features. At 40% randomly replaced neighbor entries, PreMean-GNN ROC-AUC falls from 0.96943 to 0.96773, a drop of 0.00170. DAS-GNN falls from 0.96980 to 0.96912, a drop of 0.00069. Its F1 remains 0.86992 compared with 0.86232 for PreMean-GNN. The absolute scores differ from Table 2 because this is a separately executed single-seed stress protocol. The result is consistent with the design: similarity weighting can attenuate an unrelated neighbor, and the deviation channel can expose residual inconsistency instead of forcing all context into a smoothed embedding.

6.4 Scalability and Resource Use

Table 3 and Figure 4 show measured CPU preprocessing. On the complete 257,673-node graph with 3,092,076 directed neighbor entries, graph construction takes 0.662 s and one-time propagation 0.929 s in the dedicated scaling run, totaling 1.591 s. The integer neighbor array occupies 23.59 MB and the concatenated DAS-GNN dense features occupy 121.89 MB. At 500,000 stress nodes and 6,000,000 entries, total preprocessing is 4.462 s, neighbor memory is 45.78 MB, and dense graph features are 236.51 MB. Both memory quantities scale almost exactly with N because K and feature width are fixed.

Table 3 Measured Preprocessing Scalability

Nodes	Edges	Build (s)	Prop. (s)	Cache MB
25,000	0.30 M	0.045	0.073	2.29
50,000	0.60 M	0.108	0.156	4.58
100,000	1.20 M	0.224	0.337	9.16
257,673	3.09 M	0.662	0.929	23.59
500,000*	6.00 M	2.314	2.147	45.78

Note: *The 500k row is a distribution-preserving stress graph and is not used for detection accuracy.

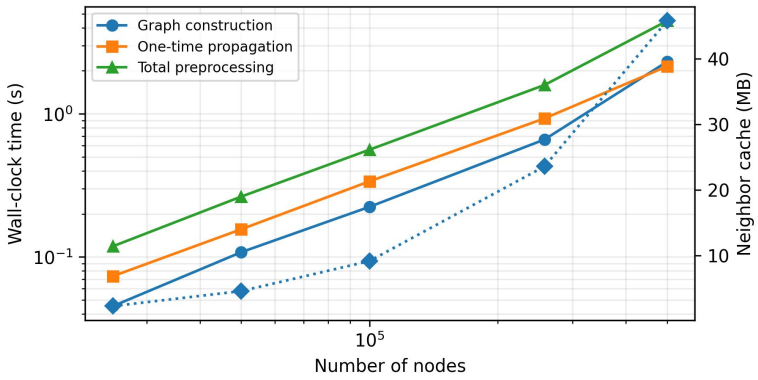

Figure 4 CPU Preprocessing Time and Integer Neighbor-Cache Growth
Note: Axes for time and nodes are logarithmic.

End-to-end neural training after precomputation averages 6.61 s for DAS-GNN, compared with 5.67 s for the MLP, on five virtual CPU cores of an AMD EPYC 9V74 host. Peak process resident memory averages 940.7 MB for DAS-GNN and 856.9 MB for the MLP. These resident-set values include Python, libraries, the full data frame, tensors, and cached arrays; they are not model-parameter memory. The 50k forward timing in the raw CSV is an operating-system scheduling outlier and is therefore not used to infer asymptotic behavior. The stable observations are cache size, total preprocessing, and full-dataset throughput.

6.5 Practical Interpretation

The results suggest a decision rule for practitioners. When a well-tuned boosted tree satisfies accuracy and governance requirements, it is difficult to justify replacing it solely for a small GNN gain that does not exist on this dataset. DAS-GNN is attractive when a neural pipeline is already required, when graph-derived explanations such as local deviation are useful, or when a bounded relation cache will be reused by several downstream models. The graph construction also offers a path for adding domain relations later: known host, user, or session links can replace or augment the induced neighbors without changing the decoupled detector interface.

6.6 Auditability and Failure Modes

The graph cache is an auditable intermediate artifact. Each row has a fixed list of neighbor indices, and the precomputed arrays can be recomputed from those indices without loading a trained model. This permits three checks before deployment: category-agreement histograms can reveal accidental cross-protocol mixing; similarity and entropy

distributions can reveal isolated or duplicated records; and the largest deviation coordinates can be inspected for unit or preprocessing errors. A conventional end-to-end sampled GNN can expose sampled subgraphs, but its effective neighborhood may change by epoch. DAS-GNN trades some adaptivity for a stable object that can be versioned with a model release.

Several failure modes follow directly from the construction. If an attacker deliberately mimics the marginal numeric profile and categorical signature of normal flows, locality disagreement may be small. If a category bucket is tiny, repeated boundary candidates may reduce neighborhood diversity. If a global distribution shift moves all nearby records together, deviation can remain low even though the entire region is novel. Finally, acquisition-order neighbors may be misleading when files are heavily shuffled. These cases motivate combining DAS-GNN with temporal drift monitoring, minimum bucket-size rules, or verified endpoint relations. No claim is made that a fixed locality graph replaces such domain evidence.

The stronger LightGBM result is itself a diagnostic. It indicates that much of the benchmark signal is available in individual flow attributes and nonlinear thresholds. The graph model's smaller incremental gain should therefore be judged against its additional capabilities—stable relational diagnostics and reusable graph context—not only against raw AUC. On a dataset whose anomalies are primarily relational, the ranking may differ, but that proposition requires new experiments rather than extrapolation from UNSW-NB15.

7 LIMITATIONS, THREATS TO VALIDITY, AND REPRODUCIBILITY

The main limitation is dataset breadth. A single benchmark cannot establish general superiority across enterprise traffic, financial graphs, or social networks. UNSW-NB15 is valuable because it provides an official split and detailed attack categories, but it is a controlled hybrid corpus rather than continuously evolving production telemetry [20]. The graph is induced from flow attributes and acquisition order, not from packet-level endpoint interactions. Thus, the experiment tests whether an induced locality graph enhances flow classification; it does not validate a claim about a native communication topology.

The evaluation is transductive with respect to unlabeled features because all official nodes participate in graph ordering and context precomputation. This setting is common when a batch of unlabeled events is available before scoring, but it can be optimistic for strict streaming deployment. An inductive variant can construct the training graph first and attach each arriving flow only to a fixed searchable index of training nodes. That variant would avoid test-test relations at the cost of maintaining an online candidate structure. It should be evaluated separately rather than assumed equivalent.

Only neural variants receive three-seed uncertainty estimates. The logistic and LightGBM results are single seeded reference runs. The graph-corruption study is also single seed. In addition, F1 depends on validation threshold selection and should not be transferred directly to another class prior. CPU wall-clock values depend on library versions, thread scheduling, and host contention. For this reason, the package includes raw CSV files, per-run scores and histories, software versions, graph statistics, and a plotting script. The dataset itself is not redistributed; the README records filenames and checksums and points to the official source.

No synthetic accuracy value is mixed with the real benchmark. Synthetic sampling is used only for the 500k scale row, where records are bootstrapped from the observed feature distribution with small numeric jitter. The paper reports this row with an asterisk and does not compute ROC-AUC or F1 on it. All other accuracy values come from actual executions over the official test labels. Reproduction may show small floating-point variation across BLAS and CPU versions, but the seed initialization occurs before model construction and the raw outputs allow direct auditing.

8 CONCLUSION

This paper presented DAS-GNN, a scalable graph neural framework for anomaly detection when large network-flow data arrive as a table rather than a native graph. A degree-capped multi-view construction avoids all-pairs search, cosine-gated aggregation estimates stable local behavior, and an explicit deviation channel preserves the signal that smoothing can erase. One-time propagation converts graph learning into reusable dense features, making subsequent training independent of edge traversal. On UNSW-NB15, DAS-GNN consistently improves a matched MLP and is more robust than uniform neighborhood averaging under random edge corruption. The strongest overall detector is nevertheless LightGBM, an important boundary on the empirical claim. Future work should evaluate strict inductive streaming, native endpoint relations, temporal drift, and multiple real production datasets while preserving the same auditability standard.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Pang G, Shen C, Cao L, et al. Deep learning for anomaly detection: A review. ACM Computing Surveys, 2021, 54(2): 38, 1-38. DOI: 10.1145/3439950.

[2] Liu F T, Ting K M, Zhou Z H. Isolation forest. In: Proceedings of the 8th IEEE International Conference on Data Mining, 2008: 413-422. DOI: 10.1109/ICDM.2008.17.

[3] Breunig M M, Kriegel H P, Ng R T, et al. LOF: Identifying density-based local outliers. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2000: 93-104. DOI: 10.1145/342009.335388.
[4] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of the International Conference on Learning Representations (ICLR), 2017.
[5] Hamilton W L, Ying R, Leskovec J. Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, 2017.
[6] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks. In: Proceedings of the International Conference on Learning Representations (ICLR), 2018.
[7] Chen J, Ma T, Xiao C. FastGCN: Fast learning with graph convolutional networks via importance sampling. In: Proceedings of the International Conference on Learning Representations (ICLR), 2018.
[8] Zou D, Hu Z, Wang Y, et al. Layer-dependent importance sampling for training deep and large graph convolutional networks. In: Advances in Neural Information Processing Systems, 2019.
[9] Chiang W L, Liu X, Si S, et al. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019: 257-266. DOI: 10.1145/3292500.3330925.
[10] Zeng H, Zhou H, Srivastava A, et al. GraphSAINT: Graph sampling based inductive learning method. In: Proceedings of the International Conference on Learning Representations (ICLR), 2020.
[11] Wu F, Souza A, Zhang T, et al. Simplifying graph convolutional networks. In: Proceedings of the 36th International Conference on Machine Learning, PMLR, 2019, 97: 6861-6871.
[12] Frasca F, Rossi E, Eynard D, et al. SIGN: Scalable inception graph neural networks. arXiv:2004.11198, 2020.
[13] Bojchevski A, Gasteiger J, Perozzi B, et al. Scaling graph neural networks with approximate PageRank. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020: 2464-2473. DOI: 10.1145/3394486.3403296.
[14] Ding K, Li J, Bhanushali R, et al. Deep anomaly detection on attributed networks. In: Proceedings of the SIAM International Conference on Data Mining (SDM), 2019: 594-602. DOI: 10.1137/1.9781611975673.67.
[15] Fan H, Zhang F, Li Z. AnomalyDAE: Dual autoencoder for anomaly detection on attributed networks. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020: 5685-5689. DOI: 10.1109/ICASSP40776.2020.9053387.
[16] Liu Y, Li Z, Pan S, et al. Anomaly detection on attributed networks via contrastive self-supervised learning. IEEE Transactions on Neural Networks and Learning Systems, 2022, 33(6): 2378-2392. DOI: 10.1109/TNNLS.2021.3068344.
[17] Dou Y, Liu Z, Sun L, et al. Enhancing graph neural network-based detectors against camouflaged fraudsters. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020: 315-324. DOI: 10.1145/3340531.3411903.
[18] Liu Y, Ao X, Qin Z, et al. Pick and choose: A GNN-based imbalanced learning approach for detection. In: Proceedings of the Web Conference, 2021: 3168-3177. DOI: 10.1145/3442381.3449989.
[19] Yuan X, Zhou N, Yu S, et al. Higher-order structure based anomaly detection on attributed networks. In: Proceedings of the IEEE International Conference on Big Data, 2021: 2691-2700. DOI: 10.1109/BigData52589.2021.9671990.
[20] Moustafa N, Slay J. UNSW-NB15: A comprehensive data set for network intrusion detection systems. In: Proceedings of the Military Communications and Information Systems Conference (MilCIS), 2015: 1-6. DOI: 10.1109/MilCIS.2015.7348942.
[21] Ke G, Meng Q, Finley T, et al. LightGBM: A highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems 30, 2017.
[22] Akoglu L, Tong H, Koutra D. Graph based anomaly detection and description: A survey. Data Mining and Knowledge Discovery, 2015, 29(3): 626-688. DOI: 10.1007/s10618-014-0365-y.
[23] Ma X, Wu J, Xue S, et al. A comprehensive survey on graph anomaly detection with deep learning. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(12): 12012-12038. DOI: 10.1109/TKDE.2021.3118815.
[24] Huang X, Yang Y, Wang Y, et al. DGraph: A large-scale financial dataset for graph anomaly detection. In: Advances in Neural Information Processing Systems 2022: 22765-22777.
[25] Liu K, Dou Y, Zhao Y, et al. PyGOD: A Python library for graph outlier detection. arXiv:2204.12095, 2022.
[26] Ying R, He R, Chen K, et al. Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018: 974-983. DOI: 10.1145/3219819.3219890.
[27] Davis J, Goadrich M. The relationship between Precision-Recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine Learning, 2006: 233-240. DOI: 10.1145/1143844.1143874.
[28] Saito T, Rehmsmeier M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLoS One, 2015, 10(3): e0118432. DOI: 10.1371/journal.pone.0118432.
[29] He H, Garcia E A. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(9): 1263-1284. DOI: 10.1109/TKDE.2008.239.
[30] Loshchilov I, Hutter F. Decoupled weight decay regularization. In: Proceedings of the International Conference on Learning Representations (ICLR), 2019.