LEAP: Layer-Persistent Edge-Confidence Auditing and Propagation for Robust Node Classification on Text-Attributed Graphs

Anonymous Author(s)

Abstract

Graph neural networks classify nodes by passing messages along the edges of an observed graph. However, the observed structure is often unreliable. Errors introduced by unreliable edges recur at each layer and accumulate as the network deepens. Existing robust methods mitigate this issue by refining the graph structure or adjusting aggregation weights. We instead propose LEAP (Layer-persistent Edge-confidence Auditing and Propagation), which keeps an explicit reliability state for each edge; all layers share this state, which is updated during training. In LEAP, each candidate edge is assigned an edge confidence and a routing risk. Both are first estimated from internal evidence in the graph and then refined by a large language model (LLM) that acts as an external auditor. The auditor inspects the most ambiguous edges and judges each one from its endpoint texts. The auditor's feedback revises edge confidence and leaves the graph topology unchanged. When internal and external evidence conflict, fusion moves toward the more conservative estimate. Experiments on four text-attributed graph benchmarks show that LEAP outperforms robust baselines under adversarial and random perturbations, and its advantage grows as perturbation strength increases.

CCS Concepts

• **Computing methodologies** → **Neural networks**; • **Information systems** → **Data mining**.

Keywords

Graph neural networks, robust node classification, text-attributed graphs, large language models, edge auditing

ACM Reference Format:

Anonymous Author(s). 2027. LEAP: Layer-Persistent Edge-Confidence Auditing and Propagation for Robust Node Classification on Text-Attributed Graphs. In *Proceedings of the ACM Web Conference 2027 (WWW '27).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Node classification is a fundamental task powering numerous applications, including citation network analysis, social media platforms, recommendation systems, and web graph learning. Graph neural networks (GNNs) address this task by propagating feature information along the observed graph topology, which enables labeled nodes to transfer discriminative signals to their unlabeled neighbors [11, 17, 33]. This mechanism implicitly assumes that the observed edges encode genuine relations, yet this assumption rarely holds in practice. Edges in practical graph datasets are mostly extracted from behavior and relational logs instead of rigorous manual labeling. Typical noise sources include perfunctory citations in citation networks, random coincident clicks in recommendation logs, unresolved entity mismatches in knowledge graphs, spurious edges in Web social and click graph, as well as a large volume of unrecorded meaningful interactions. These noisy edges produce non-local errors. As GNNs share the same adjacency matrix across all layers, misleading information from unreliable edges propagates at every message-passing hop. The resultant bias accumulates as the network deepens and spreads to nodes remotely connected to the noisy edge [49]. The problem is especially acute on text-attributed graphs (TAGs), where node texts deliver semantic cues for relational validity that topology alone fails to capture.

Existing robust GNN methods mitigate structural noise following three mainstream paradigms: purifying the graph prior to information propagation [15, 48], adjusting the aggregation under graph perturbations [19, 32], and leveraging semantic evidence extracted from large language models (LLMs) [12, 26, 37]. However, these methods assess edge reliability in a one-shot manner, decoupled from the message propagation loop. The resulting judgment is then frozen into a repaired adjacency, a set of aggregation weights, or an LLM-extracted semantic signal; it is not updated as the model learns, and the layers have no record of how reliable each edge is. Such one-shot reliability commitments bring vulnerabilities on two fronts. A noisy edge retained after pre-repair continuously propagates misleading signals across all downstream layers, whereas a mistakenly discarded valid edge is permanently excluded from message passing. Worse still, since repair statistics are estimated based on the originally contaminated graph, adversarial edges may appear locally credible exactly under the strongest attack scenarios [49]. These limitations raise a natural question: can GNNs treat edge reliability as an explicit state accessible at each layer and updatable throughout propagation, without committing to an early fixed judgment or altering the underlying graph topology?

To answer the question, we propose LEAP[1], which maintains per-edge reliability as a persistent state on TAGs. Specifically, it first builds a candidate pool from the observed edges and retrieved plausible latent non-edges, then scores each candidate via complementary internal graph signals computed from the graph. Candidates whose internal evidence is ambiguous are sent to an external LLM auditor under a small query budget. The auditor only accesses

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WWW '27,

© 2027 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-XXXX-X/2027/XX
https://doi.org/XXXXXXX.XXXXXXX

[1] Code: https://anonymous.4open.science/r/leap_code-D6B7/

endpoint text information without access to node labels or train/test splits, and its cached judgment never alters the original graph topology. Conflict-aware fusion then turns the evidence into two values for each edge: an edge confidence α_{ij} and a routing risk β_{ij}. Within a forward pass, one pair is shared by all layers, and the pair is re-fused at every training epoch as the internal evidence is refreshed. A dual-path layer references these two values at every hop: reliable feature messages are propagated with full weight, while contested ones are attenuated. Whenever internal and external evidence conflict, fusion moves toward the more conservative estimate, so that the transmission of an audited edge never exceeds its fused confidence. To summarize, our contributions are as follows.

- **Persistent edge state.** We cast robust node classification as maintaining a persistent state $(\alpha_{ij}, \beta_{ij})$ for each edge, which every GNN layer reads. We design a dual path propagation rule under which the weight of a conflicting edge is its edge confidence. .
- **Selective LLM audit.** We deploy a budget-conscious LLM policy acting as a selective external auditor, which only verifies candidate edges where external textual evidence can most effectively revise model internal judgement. Our conflict-aware fusion separates edge confidence from routing risk. Notably, the auditor is label-free and cached, and it never rewrites the graph topology.
- **Severity-adaptive superior robustness.** Evaluated across four TAG benchmarks under Metattack, LEAP achieves an average accuracy of 78.8%, outperforming the strongest baseline (77.7%). It improves over vanilla GCN (Graph Convolutional Network) by up to 26.4 percentage points under 25% structural perturbation, and consistently surpasses GCN on random edge addition/deletion, PRBCD transfer attacks and sparse label settings.

2 Related Work

We group related work into three branches: (i) structure purification methods, (ii) robust propagation methods, and (iii) methods that combine graphs with LLMs.

Structure purification decides which observed edges to trust before message passing begins. Pro-GNN alternates low-rank sparse structure optimization with GNN training [15], PTDNet learns edge-drop probabilities [24], and CoGSL and SE-GSL regularize the learned graph through mutual-information compression and structural entropy [22, 48]. Follow-up work spans joint rewiring and denoising [21], structural-noise reduction [5], poisoning sanitization [46], reliable representations [4, 20], contrastive refinement over reliable neighborhoods [47], graph reshaping [34], self-supervised purification [18], and robust condensation [7]. These methods carefully reason about topology quality, yet they produce a single repaired adjacency matrix that is frozen prior to training. As a result, any error surviving the repair propagates across every hop, and adversarial edges frequently persist, since repair statistics are calculated over the very graph contaminated by the attacker. In contrast, LEAP keeps $(\alpha_{ij}, \beta_{ij})$ as soft, independently maintained variables and does not edit the topology.

Robust propagation reweights, masks or regularizes intermediate messages during aggregation while preserving the original graph topology. PA-GNN transfers robustness from clean meta-graphs [32], SAT integrates spectral constraints with adversarial training [19], and subsequent research investigates adversarial edge dropping, unbiased aggregation, and mixed node–edge noise [3, 13, 38, 45]. Studies on heterophilic graphs further reveal that misaligned neighborhood distributions degrade aggregation performance even under benign settings [42]. Gosch et al. [10] demonstrate that attack construction and evaluation protocol can dominate robustness conclusions. Closest to our setting, GNNGuard [39] reweights and prunes edges at every layer from the similarity of the current-layer representations and blends these coefficients across layers through a layer-wise memory; and RUNG [13] recomputes edge weights from feature differences at each unrolled aggregation step. Both compute one weight per edge from hidden representations of the graph. LEAP takes a different design point: one state per edge, fixed within a forward pass, fused from graph and text evidence, and split into confidence and routing risk. Moreover, in prior methods reliability remains implicit and no trace is stored to interpret why certain edges are down-weighted after training. By contrast, LEAP keeps this record explicit.

Graph–LLM methods enrich TAG learning with external semantics. TAPE converts LLM explanations into node features [12], LLM-GNN and P2TAG use LLMs as label or prompting interfaces [2, 41], LOGIN consults an LLM during training [26], and Locle couples pseudo-labeling with rewiring [37]; a broader line of work studies topology enhancement, prompting, and graph–language interfaces [16, 23, 29, 30, 35, 43]. Most of these methods inject LLM signals at the node or training stage. LLM4RGNN [40] is the closest LLM defense. It distills GPT-4 edge judgments into a local LLM that removes malicious edges and trains a language model predictor that adds missing ones; the GNN is then trained on the repaired graph. To the best of our knowledge, LEAP is the first work to combine both evidence sources with layer-persistent reliability for each edge and explicit conflict handling. Its LLM audits a small subset of candidate edges, the verdict is fused with internal evidence into the edge state, and the topology stays unchanged.

3 Preliminaries

Let $G = (V, E, \mathbf{X}, \mathcal{T})$ be an observed TAG, where $V = \{v_1, \ldots, v_n\}$ contains n nodes, $E \subseteq V \times V$ is a possibly noisy and incomplete edge set, $\mathbf{X} \in \mathbb{R}^{n \times d}$ stacks d-dimensional node features, and $\mathcal{T} = \{t_i\}_{i \in V}$ stores the node texts. A small training subset $V_{\text{tr}} \subset V$ carries labels $\mathbf{Y}_{\text{tr}}$ over C classes. The task is robust semi-supervised node classification: predict the labels of $V \setminus V_{\text{tr}}$ accurately even when the observed structure is perturbed.

A standard GNN with L layers updates, at layer ℓ,

$$\mathbf{h}_i^{(\ell+1)} = \phi^{(\ell)}\left(\mathbf{h}_i^{(\ell)}, \sum_{j \in \mathcal{N}(i)} \psi^{(\ell)}(\mathbf{h}_i^{(\ell)}, \mathbf{h}_j^{(\ell)})\right)$$

with $\mathbf{h}_i^{(0)} = \mathbf{x}_i$, where $\mathbf{x}_i$ is the i-th row of $\mathbf{X}$, $\mathcal{N}(i)$ is the observed neighborhood of node i, and $\psi^{(\ell)}$ and $\phi^{(\ell)}$ are the message and update functions. Each GNN layer reuses the same neighborhood, so errors from unreliable edges propagate across layers and accumulate with depth.

4 Our Method

4.1 Overview

The analysis in Section 1 suggests two requirements for a robust design. First, edge reliability should be judged from more than one evidence source. For example, graph-internal statistics computed on the perturbed graph inherit the noise that needs to be detected; endpoint texts are untouched by structural attacks, so they provide an independent view. Second, even decisions informed by both sources should remain revisable during training, as one-shot fixed decisions propagate errors across all layers.

Guided by these requirements, we design LEAP. It fuses the two views into a state for each edge that every layer reads and that is updated as training proceeds.

Figure 1 shows our method LEAP from a higher level. It runs in four stages: (1) Candidate edge construction fixes the edges that may carry messages, including both observed edges and plausible missing links. An internal scorer then gives each candidate a soft routing verdict from internal signals alone. (2) External LLM auditing then queries an LLM, under a small budget, about the edges that internal evidence leaves ambiguous. (3) Conflict-aware fusion combines both verdicts into a persistent state for each edge. (4) Dual path propagation reads that state at every GNN layer.

The stages exchange only the quantities needed downstream. The internal scorer maps the evidence vector $\mathbf{z}_{ij}^{\text{in}}$ to a confidence–routing–uncertainty triplet $[c, \tau, u]_{ij}^{\text{in}}$; the superscripts in and out distinguish internal (graph) and external (LLM) evidence. A valid audit supplies the corresponding external triplet. Fusion then writes the persistent pair $(\alpha_{ij}, \beta_{ij})$. Propagation uses α_{ij} as edge confidence and β_{ij} as routing risk. For validly audited edges, β_{ij} captures disagreement and shared uncertainty between the two sources; for the other edges, it reflects the remaining internal uncertainty.

4.2 Candidate Edge Construction

Noise in the observed graph appears as spurious edges and missing links. To handle both, we begin by fixing the set of candidate edges allowed to participate in propagation. For each node $i \in V$, labeled or not, we rank its non-neighbors by a value v_λ,

$$v_\lambda = \lambda_x \cos(\mathbf{x}_i, \mathbf{x}_j) + (1 - \lambda_x) \frac{|\mathcal{N}(i) \cap \mathcal{N}(j)|}{|\mathcal{N}(i) \cup \mathcal{N}(j)| + \epsilon}. \tag{1}$$

Here $j \in V \setminus (\mathcal{N}(i) \cup \{i\})$, $\lambda_x \in [0, 1]$ balances the two terms, and ϵ prevents division by zero. The cosine term finds pairs with related content but no link. The Jaccard term favors pairs that share local structure. For each node we keep the top k pairs, where k is the pool size. The symmetrized union of these pairs forms E^+. Retrieval runs over all nodes and uses only node features and the observed structure; it uses no labels and no data splits. The candidate set is $\mathcal{E}_c = E \cup E^+$, where $\mathcal{N}_c(i)$ denotes the neighbors of node i in $\mathcal{E}_c$.

4.3 Internal Evidence Scoring

Each candidate edge is first judged from the graph itself, through an internal evidence vector $\mathbf{z}_{ij}^{\text{in}}$ containing five signals. (i) *Feature similarity* is the cosine of the endpoint features. (ii) *Neighborhood overlap* is the Jaccard term of Equation (1). (iii) *Label agreement* is 1 if both endpoints are training nodes of the same class, 0 if their classes differ, and 0.5 if either endpoint is unlabeled. (iv) *Role similarity* is the cosine between the endpoint descriptors [log(1 + degree), clustering, PageRank]. (v) *Anomaly score* measures how much the edge pushes its endpoints away from their local neighborhoods, and is derived by Equation (2) below.

Write $\mathbf{h}_i = \mathbf{h}_i^{(L)}$ for the final-layer embedding from the previous epoch, detached from the computation graph, and let $\bar{\mathbf{h}}_a$ and $\bar{\mathbf{h}}_a^{(-b)}$ denote the mean embeddings of the candidate neighbors of a with and without b, respectively. Then

$$\begin{cases} \Delta_{a \leftarrow b} = \|\mathbf{h}_a - \bar{\mathbf{h}}_a\|_2 - \|\mathbf{h}_a - \bar{\mathbf{h}}_a^{(-b)}\|_2, \\ \text{anom}_{ij} = \text{Norm}_{\mathcal{E}_c}\left[\frac{1}{2}\left(\Delta_{i \leftarrow j} + \Delta_{j \leftarrow i}\right)\right]. \end{cases} \tag{2}$$

Here $\Delta_{a \leftarrow b}$ is the change in node a's neighborhood discrepancy when neighbor b is included, and $\text{Norm}_{\mathcal{E}_c}$ is min–max normalization over candidate edges. For an empty excluded neighborhood, we set $\bar{\mathbf{h}}_a^{(-b)} = \mathbf{h}_a$, giving zero excluded-neighborhood distance. The embeddings and anomaly scores are refreshed every epoch. The score is a constant input within each forward pass since the embeddings are detached. A large score means the pair makes both endpoints less consistent with their neighborhoods.

A lightweight multilayer perceptron MLP_{in} maps the evidence to a three-way routing distribution over supportive, neutral, and risky, from which we read confidence and normalized entropy:

$$\begin{cases} \tau_{ij}^{\text{in}} = \text{softmax}\left(\text{MLP}_{\text{in}}(\mathbf{z}_{ij}^{\text{in}})\right) \in [0, 1]^3, \mathbf{1}^\top \tau_{ij}^{\text{in}} = 1, \\ c_{ij}^{\text{in}} = \tau_{ij,\text{sup}}^{\text{in}} + \frac{1}{2}\tau_{ij,\text{neu}}^{\text{in}}, \\ u_{ij}^{\text{in}} = -\frac{1}{\log 3} \sum_{r \in \{\text{sup,neu,risk}\}} \tau_{ij,r}^{\text{in}} \log \tau_{ij,r}^{\text{in}}. \end{cases} \tag{3}$$

The factor $1/\log 3$ scales u_{ij}^{in} into $[0, 1]$. The triplet τ_{ij}^{in} retains routing type, c_{ij}^{in} compresses it into scalar edge confidence, and u_{ij}^{in} measures ambiguity. The routing states are learned from the classification objective and audit consistency, without clean-edge annotation.

4.4 External LLM Auditing

Since internal evidence is computed on the perturbed graph, it inherits the graph's noise. To obtain an independent view, LEAP sends a small fraction of candidate edges to an external LLM that reads only the endpoint texts. Query priority combines internal ambiguity, proximity to the confidence boundary, inverse endpoint degree, and a bonus for retrieved edges:

$$\begin{aligned} \pi_{ij} = {} & u_{ij}^{\text{in}} + \lambda_q\left(1 - |2c_{ij}^{\text{in}} - 1|\right) \\ & + \lambda_v \text{Norm}_{\mathcal{E}_c}\left(\frac{1}{\sqrt{(1 + |\mathcal{N}_c(i)|)(1 + |\mathcal{N}_c(j)|)}}\right) \\ & + \lambda_v \eta_+ \mathbf{1}[(i, j) \in E^+]. \end{aligned} \tag{4}$$

Here λ_q weights the boundary term, λ_v weights the degree term, and η_+ scales the retrieval bonus. The degree term favors sparse regions, where one wrong edge has greater local impact. The query set Q contains the $\lceil r_{\text{audit}}|\mathcal{E}_c|\rceil$ edges with largest π_{ij}, where r_{audit} is the audited fraction.

For each queried pair, the LLM receives the two endpoint texts (truncated title and abstract) and whether the edge was observed or retrieved. For example, a retrieved pair between papers on "neural network pruning" and "model compression" is presented as:

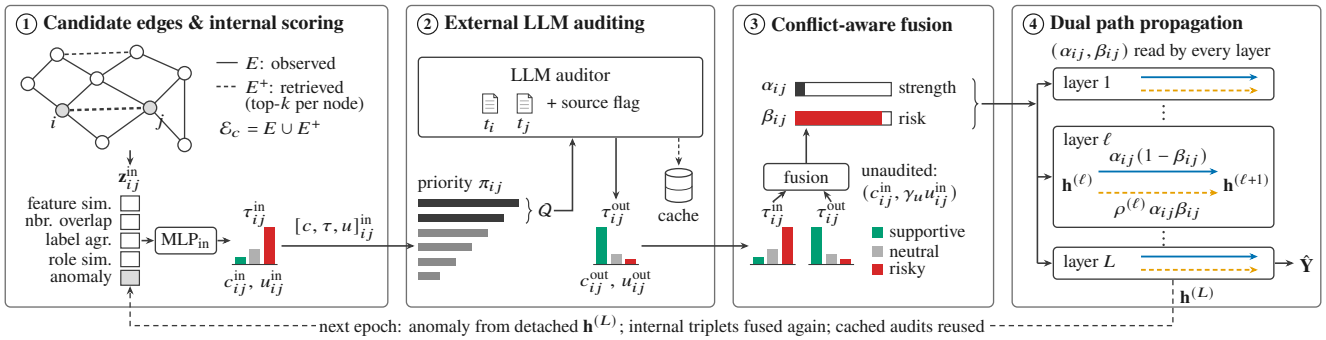


Figure 1: Illustration of our method from a higher level. Bars show the routing distributions (supportive, neutral, risky) of an example retrieved edge (i, j) on which the two sources disagree. All layers read the same $(\alpha_{ij}, \beta_{ij})$ in a forward pass, and the dashed arrow marks the refresh of the internal evidence in each epoch.

`Candidate (retrieved): [node i text]; [node j text]. Verdict: supportive, neutral, or risky?` The LLM returns a verdict with a score for each category and a short rationale. A parser normalizes valid responses into the shared triplet space, yielding τ_{ij}^{out}, c_{ij}^{out}, and u_{ij}^{out} as in Equation (3). The subset with valid responses is $\mathcal{Q}_{\text{valid}} \subseteq \mathcal{Q}$. Edges with failed responses are treated as unaudited (Section 4.5).

Every response is cached, so repeated runs (seeds, ablations, sensitivity sweeps) reuse the same verdicts and send no new queries.

4.5 Conflict-Aware Fusion

After auditing, an edge may have two sources that agree or conflict, or only the internal source. LEAP turns each case into the persistent pair $(\alpha_{ij}, \beta_{ij})$. Unaudited edges keep the internal verdict. A missing audit counts as missing evidence, not as agreement.

For valid audits, the reliability-weighted confidence is:

$$\bar{c}_{ij} = \frac{(1 - u_{ij}^{\text{in}})c_{ij}^{\text{in}} + (1 - u_{ij}^{\text{out}})c_{ij}^{\text{out}}}{2 - u_{ij}^{\text{in}} - u_{ij}^{\text{out}}}. \tag{5}$$

When its denominator is at most ϵ, we instead use $\bar{c}_{ij} = \frac{1}{2}(c_{ij}^{\text{in}} + c_{ij}^{\text{out}})$. For $(i, j) \in \mathcal{Q}_{\text{valid}}$, fusion directly combines scalar and triplet disagreement:

$$\begin{cases} \beta_{ij} = \frac{1}{2}\left(|c_{ij}^{\text{in}} - c_{ij}^{\text{out}}| + 1 - \langle \tau_{ij}^{\text{in}}, \tau_{ij}^{\text{out}} \rangle\right), \\ \alpha_{ij} = (1 - \beta_{ij})\bar{c}_{ij} + \beta_{ij} \min\{c_{ij}^{\text{in}}, c_{ij}^{\text{out}}\}. \end{cases} \tag{6}$$

In contrast, for $(i, j) \notin \mathcal{Q}_{\text{valid}}$, fusion uses $(\alpha_{ij}, \beta_{ij}) = (c_{ij}^{\text{in}}, \gamma_u u_{ij}^{\text{in}})$, where $\gamma_u \in [0, 1]$ scales residual uncertainty into routing risk. When the two sources agree confidently, β_{ij} stays small; when they conflict, β_{ij} rises and α_{ij} shifts toward the more pessimistic source. Note that β_{ij} is not a pure disagreement measure. If $\tau_{ij}^{\text{in}} = \tau_{ij}^{\text{out}} = \tau$, then $\beta_{ij} = \frac{1}{2}(1 - \|\tau\|_2^2)$, which vanishes only for a one-hot verdict; two uniform triplets give $\beta_{ij} = 1/3$. For audited edges, β_{ij} therefore measures *joint disagreement and uncertainty*, that is, how far the two sources are from a confident common verdict, and an edge on which both sources are unsure is also damped. The internal uncertainty u_{ij}^{in} is a single-source entropy, and it enters β_{ij} directly only for unaudited edges. An audit can lower an edge's influence but cannot override graph evidence with a hard edit. Concretely, α_{ij} sets how strongly an edge contributes to propagation, and β_{ij} sets how much of that contribution is routed through the attenuated conservative path. A high-α, low-β edge propagates at full strength; a high-α, high-β edge is damped rather than removed; a low-α edge is simply weak.

4.6 Dual Path Propagation

The fused pair $(\alpha_{ij}, \beta_{ij})$ above then enters the GNN through two paths: reliable messages travel at full strength, while contested messages are damped along the conservative path.

$$\begin{aligned} \mathbf{h}_i^{(\ell+1)} = \phi^{(\ell)}\Big(\mathbf{h}_i^{(\ell)}, & \sum_{j \in \mathcal{N}_c(i)} \alpha_{ij}(1 - \beta_{ij})\psi^{(\ell)}(\mathbf{h}_i^{(\ell)}, \mathbf{h}_j^{(\ell)}), \\ & \sum_{j \in \mathcal{N}_c(i)} \rho^{(\ell)} \alpha_{ij}\beta_{ij}\psi^{(\ell)}(\mathbf{h}_i^{(\ell)}, \mathbf{h}_j^{(\ell)})\Big), \end{aligned} \tag{7}$$

where $\rho^{(\ell)} \in [0, 1]$ is the conservative damping. The reliable path is dominated by high-confidence, low-risk edges; the conservative path damps contested edges and keeps them in the graph. The quantity

$$w_{ij}^{(\ell)} = \alpha_{ij}\left[1 - \beta_{ij}(1 - \rho^{(\ell)})\right] \le \alpha_{ij}, \tag{8}$$

is the total pre-normalization routing mass of edge (i, j), which never exceeds α_{ij}. For a GCN backbone, the raw weights $\alpha_{ij}(1 - \beta_{ij})$ and $\alpha_{ij}\beta_{ij}$ are symmetrized and normalized separately into $\widehat{\mathbf{A}}^{\text{rel}}$ and $\widehat{\mathbf{A}}^{\text{con}}$, giving $\mathbf{H}^{(\ell+1)} = \sigma(\widehat{\mathbf{A}}^{\text{rel}}\mathbf{H}^{(\ell)}\mathbf{W}_{\text{rel}}^{(\ell)} + \rho^{(\ell)}\widehat{\mathbf{A}}^{\text{con}}\mathbf{H}^{(\ell)}\mathbf{W}_{\text{con}}^{(\ell)} + \mathbf{H}^{(\ell)}\mathbf{W}_{\text{self}}^{(\ell)})$. The damping multiplies the *normalized* conservative path, because a uniform factor applied before symmetric normalization would cancel.

4.7 Training

The edge states and the GNN are trained jointly in an unsupervised way:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda_1\mathcal{L}_{\text{conf}} + \lambda_2\mathcal{L}_{\text{prior}} + \lambda_3\mathcal{L}_{\text{stab}}, \tag{9}$$

where $\mathcal{L}_{\text{cls}}$ is the cross-entropy loss on the training nodes V_{tr}. The three regularization terms serve distinct purposes. Over valid audits,

$$\mathcal{L}_{\text{conf}} = \frac{1}{|\mathcal{Q}_{\text{valid}}|+1} \sum_{(i,j) \in \mathcal{Q}_{\text{valid}}} \left[(1-\beta_{ij})(c_{ij}^{\text{in}} - c_{ij}^{\text{out}})^2 + \beta_{ij}\alpha_{ij}^2\right], \tag{10}$$

it aligns the two sources where they agree and penalizes high confidence under measured conflict; the unit offset in the normalizer prevents division by zero, so the term can still be computed even when no audit is valid. The source prior $\mathcal{L}_{\text{prior}} = \frac{1}{|\mathcal{E}_c|}\sum_{(i,j)}(\alpha_{ij} - \mathbf{1}[(i,j) \in E])^2$ prevents indiscriminate shrinkage: observed edges keep their weight unless evidence lowers it, while retrieved edges start weak and gain weight only with evidence. The stability term $\mathcal{L}_{\text{stab}} = \frac{1}{|V|}\sum_i \|\mathbf{h}_i^{(L)}\|_2^2$ regularizes representations at the final layer L. Training proceeds in three phases: (i) an internal warm-up of T_0 epochs with α_{ij}=c_{ij}^{in} and β_{ij}=$\gamma_u u_{ij}^{\text{in}}$; (ii) a single external audit round that fills the cache; and (iii) T epochs of joint training, in which internal triplets are refreshed and re-fused every epoch.

5 Experiments

Our experiments are organized around four research questions (RQs). **RQ1**: Does layer-persistent edge confidence improve robustness as perturbation severity grows? **RQ2**: Which components of LEAP drive the performance gain? **RQ3**: Under which settings does LEAP remain effective? **RQ4**: What do the edge decisions look like in practice?

5.1 Experimental Setup

Datasets. We use four TAG benchmarks: Cora (2,708 nodes; 7 classes), CiteSeer (3,327 nodes; 6 classes), PubMed (19,717 nodes; 3 classes) [27], and DBLP (17,716 nodes; 4 classes) [31].

Baselines. We compare the following methods: vanilla backbones (GCN [17] and GAT [33]), classic robust GNNs (RGCN [44], GCN-Jaccard [36], GCN-SVD [6]), purification and graph structure learning (Pro-GNN [15], PTDNet [24], CoGSL [22], JDR [21], GRANCE [47]), robust propagation (PA-GNN [32], SAT [19], GNNGuard [39], RUNG [13], ADEdgeDrop [3], GNN Defender [38]), the LLM defense LLM4RGNN [40], and RSGNN [4]. As for our method LEAP, we also design two variants called LEAP-int and LEAP-SBERT, respectively. The former only uses the internal evidence (i.e., r_{audit}=0), while the latter replaces the LLM auditor with a Sentence-BERT (SBERT) similarity auditor.

Evaluation Metrics. Following prior work [4, 19, 21], we use accuracy to measure the performance. Accuracy is reported as the mean and sample standard deviation over ten seeds. To examine the edge decisions themselves, we employ several other metrics including *coverage*, the *high-β* fraction, *agreement*, and the external confidence $conf_{\text{out}}$. Moreover, we also employ the AUROC for separating injected from clean edges. More details about evaluation metrics and experiment reproducibility please refer to Appendix A.

5.2 Main Results (RQ1)

We first examine robustness under the most common attack. Table 1 reports results under untargeted Metattack [49], a poisoning attack based on meta gradients, at three perturbation rates.

The advantage of LEAP grows with perturbation severity, answering RQ1 positively. Further, we can see that LEAP ranks best on three of the four datasets at 25% perturbation; and its average over the twelve cells (4 datasets × 3 perturbations) is 1.1 points above that of the strongest baseline GNN Defender (77.7). These results very the superiority and competitiveness of LEAP.

Averaged over the four datasets, LEAP is on par with the strongest baseline at 5% perturbation (80.3 versus 80.5 for GNN Defender), and leads it by 2.1 points at 25% perturbation (77.3 versus 75.2). The reason could be that, a one-shot repair fixes every edge decision in advance, a misjudgment (that survives the repair) adds bias at every hop. LEAP instead reads $(\alpha_{ij}, \beta_{ij})$ at every layer, revises the pair across epochs, and damps contested edges while keeping them.

From this table we also observe that both LEAP and LEAP-SBERT outperform LEAP-int in each cell, and also on average (by 3.8 and 3.1 points, respectively). Recall Section 5.1, LEAP-int has no the external audit. This essentially demonstrates the advantages of the external audit. Note that, the gain does not rely on an LLM alone. This can be understood by comparing LLM4RGNN with LEAP. Both methods employs an LLM, but LLM4RGNN uses it to purify the graph once before training. As a result, at 25% perturbation, LEAP exceeds it by 2.9, 3.2, and 2.9 points on Cora, CiteSeer, and DBLP.

To examine whether the Metattack trend extends to other settings, we also test four more protocols, the results are shown in Table 2. Relative to GCN, the gain ranges from 1.1 to 12.2. This set of results demonstrates the effectiveness of our method across attack protocols, and also shows that various attack protocols may affect the performance.

5.3 Ablation Study (RQ2)

To study the effect of internal signals, we remove each internal signal in turn from the evidence vector, as shown in Table 3 (a). We can see that every removal lowers accuracy, and the anomaly score matters most: without the anomaly score, accuracy drops by 2.6 points. No other signal costs more than 1.7 points. Even the weakest variant stays more than 23 points above GCN, so no single signal carries the robustness alone.

We next check whether the remaining gain comes from the texts themselves. To this end, we swap the external auditor while keeping others fixed, as shown in Table 3 (b). The SBERT auditor improves over LEAP-int, yet GLM-5.1 stays ahead of it. Both auditors read the same endpoint texts, so the gap comes from how they read them: SBERT scores a pair by the cosine similarity of the two text embeddings, whereas the LLM judges whether the relation should carry messages. In addition, the table also reports the results of Llama-3-8B-Instruct, Qwen3-8B, and GPT-5-mini. Every LLM auditor improves over LEAP-int by at least 3.4 points on average, implying that the gain does not depend on one particular LLM, although different LLMs may achieve various gains. Generally, internal evidence supplies most of the robustness, while the external audit corrects it if the evidence is ambiguous.

We next compare the variants of the edge state while keeping the internal evidence and the GLM-5.1 audit fixed, in order to check how it is formed and used. Table 3 (c) reports the results, which show that every variant falls below the full model in all three cells, and each edge state exerts measurable effect on performance. For example, Freezing the state costs 2.6 points, implying that refreshing it across epochs is useful.

Table 1: Node classification accuracy (%, mean±std) under untargeted Metattack at 5%/15%/25% perturbation (Pro-GNN split, 2-layer GCN backbone). Best and second best per column; "Avg." is the mean over the twelve dataset–rate cells.

	Cora			CiteSeer			PubMed			DBLP			
Method	5%	15%	25%	5%	15%	25%	5%	15%	25%	5%	15%	25%	Avg.
GCN	79.2±0.3	67.6±0.6	52.1±0.2	73.0±0.3	66.3±1.1	58.5±0.5	83.3±0.3	78.5±0.4	74.5±0.1	73.3±0.7	68.1±0.2	57.1±0.8	69.3
GAT	78.6±1.0	67.7±1.8	52.6±2.4	72.4±2.4	67.9±1.6	60.6±2.3	81.8±0.6	77.0±0.7	72.6±1.2	77.8±2.2	68.6±2.8	51.5±2.1	69.1
RGCN	77.4±0.4	68.8±0.4	57.5±0.8	70.5±0.4	65.7±0.4	55.4±0.7	81.1±0.2	73.9±0.3	68.0±0.2	72.0±0.2	66.4±1.3	58.2±0.8	67.9
GCN-Jaccard	79.1±0.6	71.0±0.6	60.8±1.1	70.5±1.0	66.0±0.9	59.9±1.5	86.4±0.1	84.8±0.1	83.7±0.1	71.3±1.4	65.7±1.8	58.3±1.6	71.5
GCN-SVD	78.4±0.5	66.7±1.2	52.1±1.2	68.8±0.7	63.3±1.0	57.2±1.9	83.4±0.2	83.1±0.2	82.7±0.2	80.1±0.8	65.1±0.9	57.6±0.4	69.9
Pro-GNN-fs	81.8±0.4	76.0±1.1	56.5±2.6	73.1±0.3	70.8±0.9	66.4±2.6	87.2±0.1	86.0±0.1	84.7±0.1	76.0±2.2	68.9±2.7	60.4±2.1	74.0
Pro-GNN	82.3±0.5	76.4±1.3	69.7±1.7	72.9±0.6	72.0±1.1	69.0±2.8	**87.6**±0.1	86.8±0.2	85.8±0.2	77.1±2.0	70.3±2.4	61.9±1.9	76.0
PTDNet	77.4±0.5	64.2±0.4	58.5±0.5	71.9±0.4	66.4±0.2	57.9±0.9	85.4±0.1	85.0±0.2	83.2±0.1	71.6±0.8	67.0±0.2	55.8±0.5	70.4
CoGSL	79.4±0.2	68.6±0.6	60.0±2.2	75.9±0.1	70.8±0.2	61.4±0.7	85.1±0.1	81.2±0.1	78.1±0.1	68.9±0.4	60.7±2.0	56.8±1.1	70.6
PA-GNN	81.0±0.8	76.0±1.4	64.4±1.1	72.8±0.4	69.0±0.9	62.9±1.0	81.7±0.2	80.3±0.3	76.7±0.4	78.2±1.2	66.5±1.3	59.6±0.1	72.4
SAT	78.4±0.9	76.9±0.9	75.3±0.3	70.9±0.3	66.4±1.4	64.7±1.1	86.1±0.1	85.5±0.2	85.1±0.1	**82.1**±0.6	**74.4**±1.1	64.2±0.3	75.8
GNNGuard	81.0±0.4	75.5±0.7	70.5±0.8	73.8±0.5	70.9±0.5	66.9±0.6	86.5±0.3	85.3±0.2	84.1±0.2	77.6±1.1	70.4±1.0	64.7±1.3	75.6
RUNG	81.7±0.4	77.3±0.6	74.8±0.5	74.7±0.5	71.7±0.6	70.2±0.7	86.7±0.2	85.8±0.3	84.8±0.3	77.3±1.0	71.5±1.1	66.6±1.0	76.9
JDR	81.9±1.2	75.2±1.1	74.0±1.7	74.3±0.6	73.6±0.5	71.2±0.2	87.0±0.1	86.7±0.2	**86.6**±0.1	74.7±0.8	72.0±1.7	68.0±2.4	77.1
GRANCE	81.4±0.4	76.5±0.6	72.8±1.1	74.9±0.4	72.6±0.7	70.3±1.1	87.0±0.2	86.3±0.2	85.6±0.3	76.9±1.0	71.4±0.8	66.8±1.2	76.9
ADEdgeDrop	81.0±0.5	76.2±0.8	71.3±0.8	74.3±0.7	70.8±0.6	67.9±1.1	86.5±0.2	85.5±0.3	84.3±0.4	77.9±0.9	71.6±0.7	66.4±1.9	76.1
LLM4RGNN	81.5±0.5	78.6±0.7	75.6±0.2	75.0±0.7	72.9±1.0	71.0±1.0	87.1±0.3	85.9±0.3	85.1±0.3	75.4±1.4	70.7±1.4	67.6±1.4	77.2
GNN Defender	82.2±0.7	78.0±0.7	75.3±0.7	75.2±0.8	72.7±0.5	70.9±0.4	87.2±0.3	86.0±0.2	85.7±0.5	77.5±1.1	72.2±1.8	69.0±1.4	77.7
LEAP-int	81.2±1.8	79.1±0.5	73.6±0.4	74.6±0.1	70.9±0.6	66.4±0.4	86.3±0.6	83.8±0.3	80.6±0.8	74.0±2.3	67.5±0.6	61.3±0.6	74.9
LEAP-SBERT	82.2±0.5	80.9±0.6	77.5±0.4	75.8±0.6	73.8±0.6	73.3±0.6	87.0±0.3	86.3±0.3	85.0±0.2	74.4±1.0	70.5±1.0	69.0±1.4	78.0
LEAP	**82.4**±0.4	**81.6**±0.9	**78.5**±0.2	**76.3**±0.6	**74.6**±0.3	**74.2**±1.1	87.4±0.3	**86.9**±0.8	86.1±0.9	75.0±2.3	71.7±0.6	**70.5**±0.8	**78.8**

Table 2: Robustness results under various protocols (accuracy %). Gain = LEAP − GCN; the sparse label row averages 16 dataset–perturbation cells.

Protocol	Scope	GCN	LEAP	Gain
Random add	Cora +10%	80.8	**81.9**	+1.1
	CiteSeer +20%	70.5	**74.1**	+3.6
Deletion 95%	Cora	67.7	**74.3**	+6.6
	CiteSeer	67.1	**72.9**	+5.8
PRBCD transfer [9]	Cora 15%	53.2	**65.4**	+12.2
	CiteSeer 15%	57.2	**68.1**	+10.9
	PubMed 15%	74.6	**85.5**	+10.9
Sparse-label	16-cell mean	65.0	**74.0**	+9.0

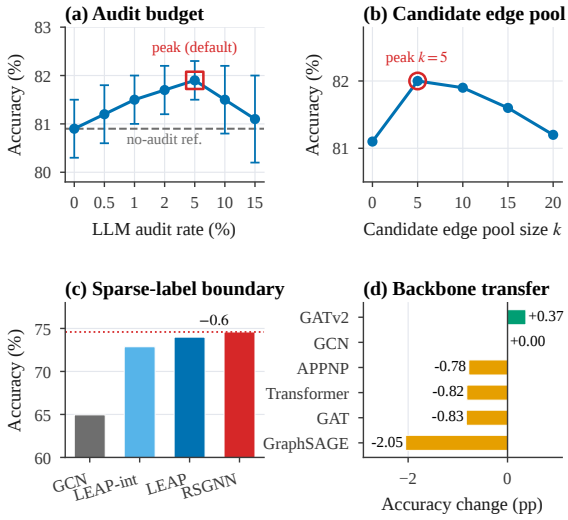


Figure 2: Sensitivity and limits. (a) Dashed: accuracy without auditing; square: the 5% peak; (b) on Cora with 10% random addition; (c) limit under sparse labels, compared with RSGNN. (d) Accuracy change from instantiating the dual path on each backbone, measured against LEAP with the default GCN backbone and averaged over the six cells of Table A-4.

5.4 Sensitivity Analysis (RQ3)

Sweeping the *audit rate* from 0 to 15% traces a single peak, as shown in Figure 2 (a). Specifically, accuracy peaks at 5%, one point above the result without auditing. Moreover, the 1~10% rates stay within one standard deviation of this peak, and accuracy drops slightly as the budget grows to 15%. This trend is consistent with the explanation suggested by the priority score in Equation (4). In other words, it ranks the most ambiguous candidate edges first, and so the earliest audits cover the cases in which external evidence can change a decision; after those cases are covered, additional queries mostly reach edges that internal evidence already handles. This implies that a small audit budget is therefore sufficient. Table 4 lists the values behind this curve; a 1% budget already gives more than half of the best gain.

Table 3: Ablations (accuracy, %; Metattack, GCN backbone). (a) Removing each internal signal on Cora under 25% Metattack; Δ is relative to the full model. (b) Replacing the auditor while keeping selection, fusion, and propagation fixed. (c) Variants of the edge state and the candidate set, with th same internal evidence and GLM-5.1 audit. The last row of each block is the full model.

(a) Internal evidence			(b) External auditor					(c) Edge state				
Configuration	Cora 25%	Δ	Auditor	Cora 15%	Cora 25%	CiteS. 25%	Avg.	Variant	Cora 15%	Cora 25%	CiteS. 25%	Avg.
GCN	52.1	−26.4	None (LEAP-int)	79.1	73.6	66.4	73.0	One-shot weights	79.0	74.4	69.3	74.2
w/o anomaly	75.9	**−2.6**	SBERT	80.9	77.5	73.3	77.2	Frozen state	79.7	75.7	71.2	75.5
w/o role sim.	76.8	−1.7	Llama-3-8B	80.2	76.8	72.1	76.4	Single path	80.8	76.9	72.6	76.8
w/o label agr.	77.3	−1.2	Qwen3-8B	80.7	77.3	73.3	77.1	Conf. only (β_{ij}=0)	80.1	75.8	70.9	75.6
w/o feature sim.	77.6	−0.9	GPT-5-mini	**81.6**	78.1	**74.3**	78.0	Risk onl (α_{ij}=1)	79.7	75.5	70.2	75.1
w/o neigh. overlap	78.0	−0.5	/	/	/	/	/	w/o retrieval	81.1	77.1	73.0	77.1
Full LEAP	**78.5**	–	GLM-5.1 (default)	**81.6**	**78.5**	74.2	**78.1**	Full LEAP	**81.6**	**78.5**	**74.2**	**78.1**

Table 4: Audit rate sweep on the Cora random edge addition cell. Values are mean ± standard deviation over ten seeds; Δ is relative to no audit.

Audit rate r_{audit} (%)	Accuracy (%)	Δ
0 (no audit)	80.9±0.6	–
0.5	81.2±0.6	+0.3
1	81.5±0.5	+0.6
2	81.7±0.5	+0.8
5 (default)	**81.9**±0.4	+1.0
10	81.5±0.7	+0.6
15	81.1±0.9	+0.2

A larger pool recovers more missing links but also admits more spurious ones. Figure 2 (b) shows both effects, and accuracy peaks at k=5. A moderate pool admits only the strongest retrieved pairs of each node. These pairs are more likely to be genuine missing links. Raising the pool to k=15 and k=20 adds pairs in the retrieval ranking, where spurious pairs are also more common. This explains why accuracy falls back by 0.3 and 0.7 points relative to k=10.

Similar to most of existing methods, LEAP bears some limitations. When LEAP is under the protocol combining structural perturbations with very few labeled nodes, it performs not good enough. For example, although LEAP improves over GCN by 9.0 points on average but remains 0.6 points below RSGNN, as shown in Figure 2 (c). This implies that LEAP repairs the propagation structure but does not replace an objective designed for sparse labels.

Furthermore, Figure 2 (d) studies the sensitivity across various backbones. We can see that the dual path stays within about two points of the default GCN backbone on average. This demonstrates that the gap across various backbones is relatively small, implying the robustness from another point of view.

5.5 Audit Diagnostics and Efficiency (RQ4)

Since every audit is cached, the edge decisions can be inspected afterwards (Figure 3; denominators in Equations (A-1)–(A-2)).

Coverage and verdicts. Audit coverage of observed edges stays at 8.6–8.9% across Metattack rates on Cora (Figure 3a). At most 3.5% of edges receive a high routing risk, most under PRBCD transfer (Figure 3b), so the conservative path leaves most edges untouched.

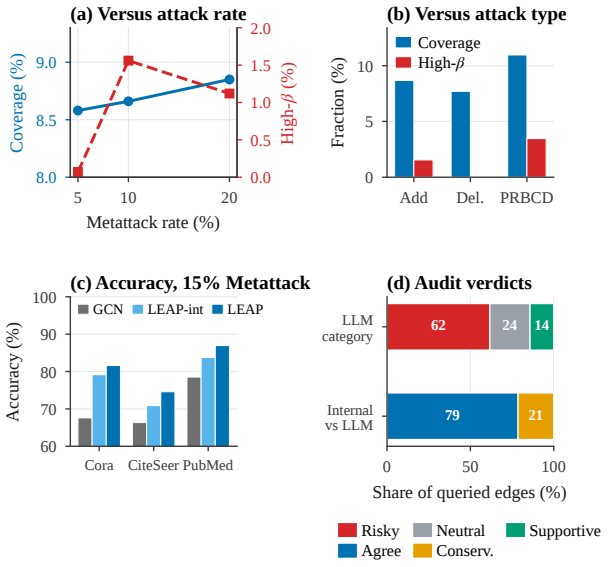


Figure 3: Diagnostics of edge decisions from cached audit records. (a) Coverage of observed edges (left axis) and fraction of candidate edges with $\beta \geq 0.9$ (right axis) versus Metattack rate on Cora. (b) The same quantities across stress types. (c) LEAP, LEAP-int, and GCN at 15% Metattack. (d) External verdict mix and agreement between internal and external categories.

Of all queries, 96.9% produce valid responses; the rest fall back to internal evidence through the validity mask. The auditor agrees with the internal scorer on 78.6% of valid records (Figure 3d). Most disagreements involve candidate edges whose endpoints share no neighbors but have semantically aligned texts. This added information raises accuracy over LEAP-int by 2.5–3.7 points (Figure 3c).

Edge detection. Table 5 evaluates the edge decisions directly. On the observed edges of each perturbed graph, it ranks edges by $1-w_{ij}$ and reports the AUROC for separating injected from clean edges; the row for feature similarity ranks edges by $1-\cos(\mathbf{x}_i, \mathbf{x}_j)$. This test does not depend on downstream accuracy, so it shows whether the routing weights themselves identify injected edges. Internal

Table 5: Detection of injected edges under Metattack (AUROC, %).

	Cora		CiteSeer	
Edge score	15%	25%	15%	25%
Feature similarity	74.2±0.7	71.7±1.2	71.5±0.8	69.2±1.0
LEAP-int ($1 - w_{ij}$)	86.0±0.8	82.4±0.9	83.0±1.3	78.5±0.9
LEAP ($1 - w_{ij}$)	**89.3**±0.7	**87.1**±0.9	**87.1**±0.6	**84.8**±1.2

Table 6: Measured training time over at most $T_0 + T$ epochs and latency per inference pass. † denotes the initial LLM audit at $r_{\text{audit}} = 5\%$.

	Cora		PubMed	
	Train (s)	Infer (ms)	Train (s)	Infer (ms)
GCN	18	4	62	14
LEAP-int	24	5	81	17
LEAP (cached)	27	5	89	18
LEAP (first run)	27 + 79†	5	89 + 270†	18

evidence alone (LEAP-int) already exceeds feature similarity by 9.3 to 11.8 points. The audit adds another 3.3 to 6.3 points, and its contribution is largest at 25%, where more injected edges look locally plausible. With the audit, the AUROC of LEAP stays above 84% in all four settings. Together, these diagnostics show that the audit outputs are parseable, informative, and complementary to the internal evidence.

Running time. Table 6 separates cached training from the initial external audit. Cached LEAP trains 1.4–1.5× slower than GCN on both datasets and is 25–29% slower per inference pass. In absolute terms, inference adds at most 4 ms per pass, and cached LEAP trains only slightly slower than LEAP-int (89 versus 81 s on PubMed), so fusing the cached audit adds little to the cost of the dual path. The first run additionally pays the audit once (79 s on Cora and 270 s on PubMed); later runs read the cache.

Scalability. Table 7 lists, for each dataset, the size of the candidate set, the number of audit queries, the processed tokens, and the running time of the single audit round. The ogbn-arxiv graph [14] (169,343 nodes; 40 classes) is about nine times larger than PubMed; Table 8 reports LEAP on it. Together, the two tables show how the audit overhead and the training time grow with graph size. The number of queries is 5% of $|\mathcal{E}_c|$ on every dataset, so the audit overhead grows linearly with the candidate set. Prompts account for over 90% of the tokens, and each query uses 760 to 1,060 tokens. On ogbn-arxiv, the candidate set is 12 times larger than on PubMed. The audit round takes 72 minutes and runs only once, because later runs read the cache. On the accuracy side, LEAP stays close to GCN on the clean graph (72.2 versus 71.8), improves over it by 1.2 points under random addition, and by 7.3 points under PRBCD, of which the audit contributes 1.7 points. Training takes 1.4× as long as GCN, the same ratio as on the smaller datasets.

Reproducibility details and additional experimental results are provided in the Appendix.

Table 7: Overhead of the initial LLM audit at r_{audit}=5%. Tokens (in millions) count prompt and response tokens; time is the running time of the audit round.

Dataset	$\lvert\mathcal{E}_c\rvert$	Queries	Tokens	Time (s)
Cora	25,179	1,259	0.956	79
CiteSeer	21,802	1,091	0.916	73
PubMed	197,388	9,870	9.660	270
DBLP	223,106	11,156	9.854	338
ogbn-arxiv	2,360,988	118,050	125.270	4,318

Table 8: Accuracy (%) and training time on ogbn-arxiv. PRBCD follows the transfer setting of Table 2.

Method	Clean	Random add 10%	PRBCD 15%	Train (s)
GCN	71.8±0.2	70.4±0.3	61.3±0.6	621
LEAP-int	71.9±0.3	71.1±0.2	66.9±0.5	772
LEAP	**72.2**±0.2	**71.6**±0.2	**68.6**±0.5	891

6 Conclusion

We recast robust node classification on TAGs as layer-persistent edge-confidence estimation. The core idea of our method is that, every candidate edge carries a persistent state $(\alpha_{ij}, \beta_{ij})$, all GNN layers read the persistent state, and it is fused again at every training epoch. Besides, a budgeted external LLM auditor that uses no labels refines edge confidence from endpoint texts alone. Experiments on four TAG benchmarks demonstrate its superiority and competitiveness. Future work will investigate adaptive attacks tailored to retrieval and auditing, and extend persistent edge states to other graph settings.

References

[1] Shaked Brody, Uri Alon, and Eran Yahav. 2022. How Attentive are Graph Attention Networks?. In *ICLR*.

[2] Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. 2024. Label-free Node Classification on Graphs with Large Language Models (LLMs). In *ICLR*.

[3] Zhaoliang Chen, Zhihao Wu, Ylli Sadikaj, Claudia Plant, Hong-Ning Dai, Shiping Wang, Yiu-Ming Cheung, and Wenzhong Guo. 2025. ADEdgeDrop: Adversarial Edge Dropping for Robust Graph Neural Networks. *IEEE Transactions on Knowledge and Data Engineering* 37 (2025), 4948–4961.

[4] Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. 2022. Towards Robust Graph Neural Networks for Noisy Graphs with Sparse Labels. In *WSDM*. 181–191.

[5] Mingze Dong and Yuval Kluger. 2023. Towards Understanding and Reducing Graph Structural Noise for GNNs. In *ICML*, Vol. 202. 8202–8226.

[6] Negin Entezari, Saba A. Al-Sayouri, Amirali Darvishzadeh, and Evangelos E. Papalexakis. 2020. All You Need Is Low (Rank): Defending Against Adversarial Attacks on Graphs. In *WSDM*. 169–177.

[7] Xinyi Gao, Hongzhi Yin, Tong Chen, Guanhua Ye, Wentao Zhang, and Bin Cui. 2025. RobGC: Towards Robust Graph Condensation. *IEEE Transactions on Knowledge and Data Engineering* 37 (2025), 4791–4804.

[8] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks Meet Personalized PageRank. In *ICLR*.

[9] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. 2021. Robustness of Graph Neural Networks at Scale. In *NeurIPS*, Vol. 34. 7637–7649.

[10] Lukas Gosch, Simon Geisler, Daniel Sturm, Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. 2023. Adversarial Training for Graph Neural Networks: Pitfalls, Solutions, and New Directions. In *NeurIPS*, Vol. 36.

[11] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*, Vol. 30. 1024–1034.

[12] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2024. Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning. In *ICLR*.
[13] Zhichao Hou, Ruiqi Feng, Tyler Derr, and Xiaorui Liu. 2024. Robust Graph Neural Networks via Unbiased Aggregation. In *NeurIPS*, Vol. 37. 110097–110130.
[14] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*, Vol. 33. 22118–22133.
[15] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph Structure Learning for Robust Graph Neural Networks. In *KDD*. 66–74.
[16] Shima Khoshraftar, Niaz Abedini, and Amir Hajian. 2025. GraphiT: Efficient Node Classification on Text-Attributed Graphs with Prompt Optimized LLMs. In *WWW Companion*. 1824–1829.
[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[18] Woohyun Lee and Hogun Park. 2025. Self-Supervised Adversarial Purification for Graph Neural Networks. In *ICML*, Vol. 267. 33715–33735.
[19] Jintang Li, Jiaying Peng, Liang Chen, Zibin Zheng, Tingting Liang, and Qing Ling. 2023. Spectral Adversarial Training for Robust Graph Neural Network. *IEEE Transactions on Knowledge and Data Engineering* 35 (2023), 9240–9253.
[20] Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. Reliable Representations Make A Stronger Defender. In *KDD*. 925–935.
[21] Jonas Linkerhägner, Cheng Shi, and Ivan Dokmanić. 2025. Joint Graph Rewiring and Feature Denoising via Spectral Resonance. In *ICLR*.
[22] Nian Liu, Xiao Wang, Lingfei Wu, Yu Chen, Xiaojie Guo, and Chuan Shi. 2022. Compact Graph Structure Learning via Mutual Information Compression. In *WWW*. 1601–1610.
[23] Jianglin Lu, Yixuan Liu, Yitian Zhang, and Yun Fu. 2025. Scale-Free Graph-Language Models. In *ICLR*.
[24] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. 2021. Learning to Drop: Robust Graph Neural Network via Topological Denoising. In *WSDM*. 779–787.
[25] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. 2024. Classic GNNs are Strong Baselines: Reassessing GNNs for Node Classification. In *NeurIPS*, Vol. 37. 97650–97669.
[26] Yiran Qiao, Xiang Ao, Yang Liu, Jiarong Xu, Xiaoqian Sun, and Qing He. 2025. LOGIN: A Large Language Model Consulted Graph Neural Network Training Framework. In *WSDM*. 232–241.
[27] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Magazine* 29 (2008), 93–106.
[28] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. 2021. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. In *IJCAI*. 1548–1554.
[29] Shengyin Sun, Yuxiang Ren, Jiehao Chen, and Chen Ma. 2026. Large Language Models as Topological Structure Enhancers for Text-Attributed Graphs. In *DASFAA*, Vol. 15991. 106–122.
[30] Yuanfu Sun, Zhengnan Ma, Yi Fang, Jing Ma, and Qiaoyu Tan. 2025. GraphICL: Unlocking Graph Learning Potential in LLMs through Structured Prompt Design. In *Findings of NAACL*. 2440–2459.
[31] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: Extraction and Mining of Academic Social Networks. In *KDD*. 990–998.
[32] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. 2020. Transferring Robustness for Graph Neural Network Against Poisoning Attacks. In *WSDM*. 600–608.
[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[34] Haibo Wang, Chuan Zhou, Xin Chen, Jia Wu, Shirui Pan, Zhao Li, Jilong Wang, and Philip S. Yu. 2024. Graph Structure Reshaping Against Adversarial Attacks on Graph Neural Networks. *IEEE Transactions on Knowledge and Data Engineering* 36 (2024), 6344–6357.
[35] Zehong Wang, Sidney Liu, Zheyuan Zhang, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. 2025. Can LLMs Convert Graphs to Text-Attributed Graphs?. In *NAACL*. 1412–1432.
[36] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. In *IJCAI*. 4816–4823.
[37] Taiyan Zhang, Renchi Yang, Yurui Lai, Mingyu Yan, Xiaochun Ye, and Dongrui Fan. 2025. Leveraging Large Language Models for Effective Label-Free Node Classification in Text-Attributed Graphs. In *SIGIR*. 698–708.
[38] Xiaobo Zhang, Yutao Liu, Hao Wang, Wei Xie, and Wei Wang. 2026. GNN Defender: Overcoming Mixed Noise Through Cross-Correction Over Nodes and Edges. *IEEE Transactions on Knowledge and Data Engineering* 38 (2026), 3655–3668.
[39] Xiang Zhang and Marinka Zitnik. 2020. GNNGuard: Defending Graph Neural Networks against Adversarial Attacks. In *NeurIPS*, Vol. 33. 9263–9275.
[40] Zhongjian Zhang, Xiao Wang, Huichi Zhou, Yue Yu, Mengmei Zhang, Cheng Yang, and Chuan Shi. 2025. Can Large Language Models Improve the Adversarial Robustness of Graph Neural Networks?. In *KDD*. 2008–2019.
[41] Huanjing Zhao, Beining Yang, Yukuo Cen, Junyu Ren, Chenhui Zhang, Yuxiao Dong, Evgeny Kharlamov, Shu Zhao, and Jie Tang. 2024. Pre-Training and Prompting for Few-Shot Node Classification on Text-Attributed Graphs. In *KDD*. 4467–4478.
[42] Xin Zheng, Yi Wang, Yixin Liu, Ming Li, Miao Zhang, Di Jin, Philip S. Yu, and Shirui Pan. 2026. Graph Neural Networks for Graphs With Heterophily: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 38 (2026), 4385–4404.
[43] Huachi Zhou, Jiahe Du, Chuang Zhou, Chang Yang, Yilin Xiao, Yuxuan Xie, and Xiao Huang. 2025. Each Graph is a New Language: Graph Learning with LLMs. In *Findings of ACL*. 17548–17559.
[44] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust Graph Convolutional Networks Against Adversarial Attacks. In *KDD*. 1399–1407.
[45] Yonghua Zhu, Lei Feng, Zhenyun Deng, Yang Chen, Robert Amor, and Michael Witbrock. 2024. Robust Node Classification on Graph Data with Graph and Label Noise. In *AAAI*, Vol. 38. 17220–17227.
[46] Yulin Zhu, Liang Tong, Gaolei Li, Xiapu Luo, and Kai Zhou. 2024. FocusedCleaner: Sanitizing Poisoned Graphs for Robust GNN-Based Node Classification. *IEEE Transactions on Knowledge and Data Engineering* 36 (2024), 2476–2489.
[47] Shuman Zhuang, Zhihao Wu, Zhaoliang Chen, Hong-Ning Dai, and Ximeng Liu. 2025. Refine then Classify: Robust Graph Neural Networks with Reliable Neighborhood Contrastive Refinement. In *AAAI*, Vol. 39. 13473–13482.
[48] Dongcheng Zou, Hao Peng, Xiang Huang, Renyu Yang, Jianxin Li, Jia Wu, Chunyang Liu, and Philip S. Yu. 2023. SE-GSL: A General and Effective Graph Structure Learning Framework through Structural Entropy Optimization. In *WWW*. 499–510.
[49] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *ICLR*.

A Reproducibility Details

A.1 Protocol, Text Preparation, Diagnostic Denominator, and External Audit

The default backbone is a 2-layer GCN with 256 hidden units, and $\mathrm{MLP}_{\mathrm{in}}$ has one 64-unit ReLU layer followed by a linear layer onto the three routing categories. The backbone study also instantiates the dual path on GraphSAGE, GAT, GATv2 [1], Graph Transformer [28], and APPNP [8, 11, 33]. Two hyperparameters are studied below: the pool size k=10 and the audit rate r_{audit}=5%. The auditor is GLM-5.1, whose responses are all cached.

Following [15], the Metattack evaluation uses the Pro-GNN protocol, that is, a 10%/10%/80% train/validation/test split on the largest connected component. Perturbations are applied before training, and their rate is the number of changed edges relative to all edges of the clean graph. We evaluate under matched robustness protocols. We enforce this rule because attack configuration alone can dominate the observed performance gaps [10, 25]. Unless stated otherwise, all attacks are poisoning attacks.

The reference loaders for Cora, CiteSeer, and PubMed read graphs, features, labels, and raw node texts from the public Graph-COM/Text-Attributed-Graphs resource. Raw strings are normalized to a *title abstract* representation before truncation. The reference package requires Python 3.10+, PyTorch 2.0+, NumPy 1.24+, and SciPy 1.10+; PyTorch Geometric is used to load the public Graph-COM files.

The external auditor evaluates the two endpoint texts together with the observed/retrieved source flag. Each request presents the endpoint titles and abstracts, defines supportive, neutral, and risky relations for message passing, and asks for three category scores plus a one-sentence rationale. Titles and abstracts are truncated to 300 and 1,400 characters, respectively. The default GLM-5.1 auditor is called through an OpenAI-compatible chat-completions endpoint with temperature 0. The alternative auditors in Table 3(b) receive the same prompt and parser.

All experiments are implemented on an NVIDIA RTX 4090 GPU, hyperparameters are shared across datasets and protocols, and all reported results average ten random seeds.

As mentioned earlier, accuracy is the main evaluation metric, and we also use other metrics. Let m_e mark a valid audit and let v_e^{in} and v_e^{out} denote the dominant internal and external categories. For the observed and retrieved candidates in $\mathcal{E}_c$, several metrics are computed as follows:

$$\begin{aligned} \text{coverage} &= \frac{\sum_{e\in E} m_e}{|E|}, \\ \text{high-}\beta &= \frac{\sum_{e\in\mathcal{E}_c} \mathbf{1}[\beta_e \geq 0.9]}{|\mathcal{E}_c|}. \end{aligned} \tag{A-1}$$

$$\begin{aligned} \text{agreement} &= \frac{\sum_{e\in Q_{\text{valid}}} \mathbf{1}[v_e^{\text{in}} = v_e^{\text{out}}]}{|Q_{\text{valid}}|}, \\ \text{conf}_{\text{out}} &= \frac{\sum_{e\in Q_{\text{valid}}} \max_r \tau_{e,r}^{\text{out}}}{|Q_{\text{valid}}|}. \end{aligned} \tag{A-2}$$

An empty denominator is reported as N/A. Coverage can exceed r_{audit} because its denominator is $|E|$, whereas $B = \lceil r_{\text{audit}}|\mathcal{E}_c|\rceil$ is defined on the larger candidate set.

Table A-1: Default hyperparameters. Adam and validation-accuracy model selection are used in every setting.

Group	Hyperparameter	Value
Candidate	pool size k	10
	retrieval balance λ_x	0.5
	numerical guard ϵ	10^{-8}
Audit	budget r_{audit}	5%
	priority weights λ_q; λ_v	0.3; 0.2
	retrieval bonus η_+	0.1
	decoding temperature	0
Fusion & prop.	uncertainty scale γ_u	0.5
	conservative damping $\rho^{(\ell)}$	0.2
Training	loss weights λ_1; λ_2; λ_3	0.5; 10^{-3}; 10^{-3}
	epochs T_0; T	50; 200
	learning rate; weight decay	5×10^{-3}; 5×10^{-4}
	stopping patience	50

A.2 Implementation Conventions

Canonical candidate edges. All graphs are treated as undirected. Self-loops are removed from the retrieval pool, and every pair is stored canonically as $(\min(i,j), \max(i,j))$. The top-k lists for all nodes from Equation (1) are unioned, symmetrized, deduplicated, and filtered against E. Thus, $E^+ \cap E = \varnothing$ and a pair proposed by both endpoints is represented once.

Internal signal conventions. Label agreement is 1 or 0 only when both endpoints belong to V_{tr}; otherwise it is set to the uninformative midpoint 0.5. Degree, clustering coefficient, PageRank, and neighborhood overlap are computed on the observed graph, whereas the anomaly signal uses the fixed candidate set $\mathcal{E}_c$. For a scalar edge statistic s, we use

$$\text{Norm}_{\mathcal{E}_c}(s_{ij}) = \frac{s_{ij} - \min_{e\in\mathcal{E}_c} s_e}{\max_{e\in\mathcal{E}_c} s_e - \min_{e\in\mathcal{E}_c} s_e + \epsilon}. \tag{A-3}$$

Table A-2: Auditor inputs, parser behavior, and cache semantics. Invalid responses use only the internal evidence through $m_{ij} = 0$.

Item	Exact protocol
Task	classify a candidate relation as supportive, neutral, or risky for message passing
Input	endpoint title/abstract strings and the observed/retrieved source flag
Output	one JSON object with a verdict, three nonnegative category scores, and a short rationale
Parser	extract JSON; read the three named scores; clip negative values; reject vectors that are not finite or that sum to zero; renormalize to the simplex
Failure	store the error status, set $m_{ij} = 0$, and use internal evidence alone
Cache key	SHA-1 of the ordered endpoint texts, source flag, and auditor model name
Stored record	edge IDs, source, normalized triplet, validity, parser status, rationale, and raw response

A constant statistic therefore maps to zero. Before entering MLP_{in}, each of the five evidence columns is standardized over $\mathcal{E}_c$ with a minimum standard deviation of 10^{-6}.

Anomaly refresh schedule. The first anomaly refresh uses a detached representation from a forward pass over the observed edges. At epoch $t > 1$, Equation (2) reads the detached final representation from epoch $t-1$; the newly computed state is then used for epoch t. If removing b leaves an empty neighborhood, $\bar{\mathbf{h}}_a^{(-b)} = \mathbf{h}_a$, as stated in the main paper. This one epoch lag defines each anomaly score before it is consumed by the current epoch.

External audit protocol. Table A-2 lists the exact auditor inputs, the parser rules, and the cache record, and Figure A-1 illustrates the interface. Each queried edge carries a validity indicator $m_{ij} \in \{0,1\}$, where $m_{ij} = 1$ only if the response is parsed into a valid triplet, so that $Q_{\text{valid}} = \{(i,j) \in Q : m_{ij} = 1\}$. The cache key does not include labels or splits.

Evidence propagation. If both source reliabilities in Equation (5) vanish, we set $\bar{c}_{ij} = (c_{ij}^{\text{in}} + c_{ij}^{\text{out}})/2$. Records with $m_{ij} = 0$ use internal evidence alone. For the GCN implementation, the raw weights $\alpha(1-\beta)$ and $\alpha\beta$ are symmetrized and normalized, separately. The factor $\rho^{(\ell)}$ multiplies the normalized conservative path, because a uniform scale applied before symmetric normalization would cancel.

Variants of Edge State In our experiments, we study several variants of edge state: (i) *One-shot weights* trains a standard GCN on $\mathcal{E}_c$ with the weights w_{ij} of Equation (8), computed once after the audit round and then fixed. (ii) *Frozen state* keeps the dual path but stops refreshing $(\alpha_{ij}, \beta_{ij})$ after that round. (iii) *Single path* sends every message through one normalized adjacency weighted by w_{ij}. (iv) *Conf. only* sets β_{ij}=0. (v) *Risk only* sets α_{ij}=1. (vi) *W/o retrieval* restricts the candidate set to the observed edges (E^+=∅).

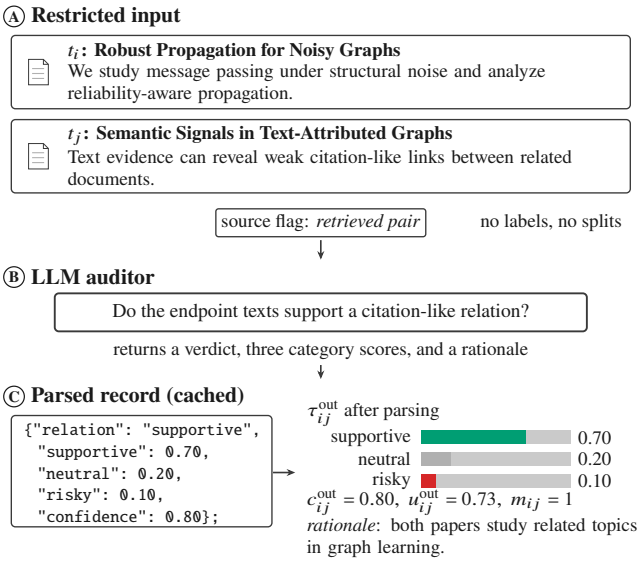


Figure A-1: The LLM audit interface. (A) The auditor receives the two endpoint texts and the source flag of the candidate edge. (B) It performs a pairwise semantic assessment from the endpoint evidence. (C) The scores for the three categories and the explanation are parsed and cached as external evidence.

B Other Experimental Results

B.1 Impact of Varying Perturbation Ratio, Datasets, and Attack Protocols

Figure A-2(a) and (b) study the trends when varying perturbation. They show that LEAP degrades much more slowly than the baselines, when the Metattack rate increases.

Figure A-2(c) studies the gain when varying datasets. We can see that the gain varies across datasets, and significantly outperforms the baseline for all these datasets. On the other hand, Figure A-2(d) shows the impact of varying attack protocols. Under PRBCD transfer, the largest observed gap occurs. This demonstrates that LEAP achieves a larger gain when stronger attacks occur.

B.2 Impact of Clean Graph, and Propagation Layers

To check whether robustness costs accuracy when the graph is clean, we conduct tests on unperturbed graphs. Table A-3(a) reports the results. We can see that LEAP exceeds GCN on all four datasets, and performs best on three out of four datasets. This implies that LEAP works well on unperturbed graphs.

Table A-3(b) reports the results of increasing the number of propagation layers L under 25% Metattack, in order to study whether the persistent state still helps when a noisy edge acts through more layers. From L=2 to L=8, GCN loses 20.6 points on Cora and 21.5 points on CiteSeer. In contrast, LEAP loses only 4.2 and 4.7 points. We conjecture that this may arise from the edge state, which continuously damps contested edges across layers.

Table A-3: Accuracy (%) on the unperturbed graphs and with various propagation layers.

(a) Unperturbed graphs

Method	Cora	CiteSeer	PubMed	DBLP
GCN	82.3±0.3	74.4±0.6	87.9±0.1	83.0±0.8
JDR	82.9±0.5	76.4±0.8	**88.4**±0.3	83.1±1.2
LEAP-int	83.0±0.5	75.7±0.5	87.7±0.2	82.8±0.5
LEAP	**83.2**±0.4	**76.9**±0.5	88.2±0.2	**83.2**±0.5

(b) Number of layers L (25% Metattack)

	Cora				CiteSeer			
Method	2	4	6	8	2	4	6	8
GCN	52.1	46.2	38.4	31.5	58.5	52.9	45.5	37.0
LEAP	78.5	78.0	75.8	74.3	74.2	74.2	72.0	69.5

Table A-4: Each backbone with and without the dual path (accuracy, %). 15% and 25% are Metattack rates; PRBCD is the 15% transfer setting of Table 2. Transformer denotes the Graph Transformer layer [28].

	Cora			CiteSeer		
Backbone	15%	25%	PRBCD	15%	25%	PRBCD
GCN	67.6	52.1	53.2	66.3	58.5	57.2
+ dual path	81.6	78.5	65.4	74.6	74.2	68.1
GAT	67.7	52.6	54.7	67.9	60.6	58.5
+ dual path	80.7	77.1	66.2	73.8	72.6	67.0
GATv2	68.9	55.8	55.4	68.1	61.7	59.3
+ dual path	82.1	79.0	65.8	75.2	73.5	69.0
GraphSAGE	68.9	53.7	54.8	67.8	59.3	57.8
+ dual path	79.8	76.0	64.0	72.9	71.5	65.9
APPNP	71.1	59.3	58.2	69.6	63.0	60.0
+ dual path	80.4	76.9	66.8	74.0	72.0	67.6
Transformer	69.7	56.7	56.3	69.0	62.4	60.0
+ dual path	81.3	78.0	63.9	74.8	73.0	66.5

B.3 Impact of Varying Backbones

Table A-4 pairs each backbone with its dual path counterpart under the same candidate set, evidence, and audit cache, at two Metattack rates and under PRBCD transfer. Each pair differs only in the propagation rule, so the gap within a pair isolates the effect of the dual path from that of the backbone. This table shows that the dual path improves every backbone in every cell, by 4.4 to 26.4 points. With the dual path, the six backbones lie within 3.1 points of one another in every column, whereas the plain backbones differ by up to 7.2 points. The robustness therefore comes mainly from the dual path rather than from a particular backbone.

B.4 Imact of Fusion strategies.

We compare conservative fusion with three alternatives while keeping the query set fixed: (i) *weighted average* where $\alpha_{ij} = \bar{c}_{ij}$ and $\beta_{ij} = 0$; (ii) *max confidence* where $\alpha_{ij} = \max(c_{ij}^{\text{in}}, c_{ij}^{\text{out}})$ and $\beta_{ij} = 0$; and (iii) *internal only* where $(\alpha_{ij}, \beta_{ij}) = (c_{ij}^{\text{in}}, \gamma_u u_{ij}^{\text{in}})$. Table A-5 shows the results.

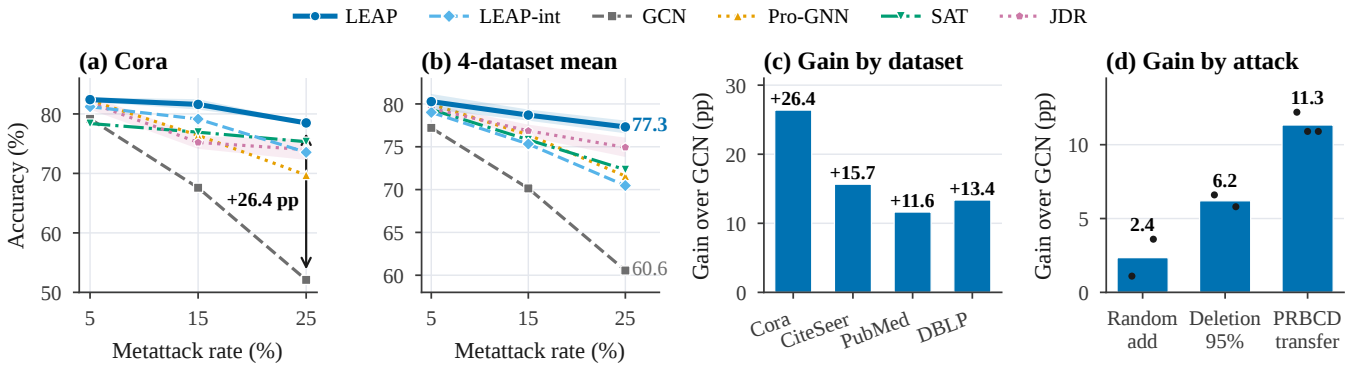


Figure A-2: Robustness under structural perturbations. (a) Accuracy versus Metattack rate on Cora; (b) the corresponding average over four datasets; (c) gain of LEAP over GCN for each dataset at 25%; (d) gain over GCN across attack families; bars are family means, and dots are dataset cells.

Table A-5: Results of Various Fusion Strategies.

	Cora		CiteSeer	
Fusion strategy	15%	25%	15%	25%
Internal only	79.1±0.5	73.6±0.4	70.9±0.6	66.4±0.4
Weighted average	80.1±0.7	75.9±0.5	72.4±0.8	70.3±0.6
Max confidence	79.8±0.6	75.2±0.4	71.8±0.7	69.5±0.5
Conservative (ours)	**81.6**±0.9	**78.5**±0.2	**74.6**±0.3	**74.2**±1.1

Conservative fusion exceeds weighted averaging by 1.5–3.9 points and max confidence by 1.8–4.7 points. The larger differences occur in the two 25% cells. Two conclusions follow. First, every fusion rule that uses the audit beats the internal-only variant, so the textual evidence itself is useful. Second, max confidence is the weakest of the three rules in all four cells. Trusting the more optimistic source lets a confident but wrong verdict keep an injected edge, which is the failure that conservative fusion is designed to prevent.