DEFENCE: ROBUST NODE CLASSIFICATION UNDER JOINT LABEL-STRUCTURE NOISE

Anonymous authors
Paper under double-blind review

ABSTRACT

How can we accurately classify nodes in graphs when both labels and edges are corrupted? Node classification under joint label and structure corruption is important in applications such as fraud detection and citation analysis. For example, in citation networks, errors in automated paper tagging may assign incorrect labels to papers, while errors in reference extraction may introduce spurious links between them. However, most prior works tackle only one noise source, over-trusting structure for label noise and labels for structure noise; when both occur, these assumptions amplify each other's errors. In this paper, we propose DEFENCE, a robust node classification model under joint label-structure noise. DEFENCE constructs anchors from pre-propagation embeddings, avoiding reliance on noisy edges and labels. These anchors regularize representation learning and predictions under joint noise, while the resulting embeddings are periodically reclustered to refine the anchors. This mutual refinement is realized through anchor-filtered contrastive learning, anchor-posterior alignment, and gradient-matched supervision. We theoretically show that even slightly better-than-random anchors effectively regularize both embeddings and predictions. Experiments on standard benchmarks demonstrate consistent improvements over strong baselines.

1 INTRODUCTION

Given a partially labeled graph whose node labels and edges are both corrupted, how can we accurately classify its nodes? Such joint corruption is common in practice (Zhong et al., 2019). In fraud detection, labels can be noisy because of delayed verification, and malicious actors can create artificial connections that introduce spurious edges (Hooi et al., 2016; Dal Pozzolo et al., 2018). In citation networks, automated topic tagging often misclassifies papers, and reference parsing can miss citations, resulting in missing links (Sauvayre, 2022).

A main challenge is that the errors from structure and labels reinforce each other. In standard GNNs, neighborhood aggregation and feature transformation are tightly coupled (Hamilton et al., 2017; Xu et al., 2019; Jin et al., 2020). When the structure is corrupted, aggregation mixes spurious neighbor signals, so the transformation is learned from distorted inputs. This distortion propagates to deeper layers and accumulates over iterations. When labels are also noisy, supervision cannot correct the drift and instead injects misleading gradients, further intensifying the propagation error. However, prior works rarely consider both noises together: label-noise methods assume the graph structure is reliable (Li et al., 2021; NT et al., 2019; Zhang et al., 2020), while edge-noise methods assume labels are reliable (Runwal et al., 2022; Ye & Ji, 2023; Dai et al., 2022). Recent works consider both in a unified loop (Zhu et al., 2024; Zhang et al., 2026), but the learning signals remain tightly coupled, so errors in one still reinforce the other.

We propose DEFENCE (DECOUPLED FEATURE ANCHORS FOR ROBUST NODE CLASSIFICATION), a robust node classification model under joint label-structure noise. DEFENCE disentangles feature embedding from graph propagation and constructs anchors by clustering the resulting pre-propagation embeddings, without directly using observed edges or labels. These noise-resilient anchors reduce the influence of corrupted edges and labels during representation learning. The updated embeddings are then used to reconstruct more accurate anchors. By repeating anchor construction and anchor-guided learning, DEFENCE jointly refines node embeddings, predictions, and anchors. This process is implemented through anchor-filtered contrastive learning (Equation (3) in Section 3.2), anchor-posterior

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

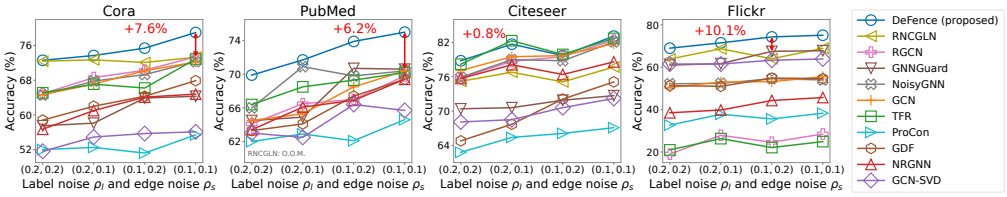

Figure 1: Node classification accuracy of the proposed DEFENCE and baselines under different ratios of label noise (ρ_l) and structure noise (ρ_s). DEFENCE outperforms the baselines in most settings.

alignment (Equation (4) in Section 3.2), and gradient-matched supervision (Section 3.3). Under better-than-random anchor assumptions, we theoretically show that anchor filtering improves class discrimination, while posterior alignment selectively regularizes predictions that follow corrupted labels (Section 3.4). Our contributions are summarized as follows:

- **Method.** We propose DEFENCE, a robust node classification method that constructs feature-derived anchors prior to propagation and exploits them as a noise-resilient reference to regularize both representation learning and posterior prediction under joint structure and label noise.
- **Theory.** We show that, under mild better-than-random conditions, anchors provide effective regularization at both the embedding and prediction levels.
- **Experiments.** We conduct various experiments on real-world datasets. DEFENCE achieves the best performance across datasets and label/structure noise combinations, as shown in Figure 1.

The symbols used in this paper are summarized in Table 3. The code and datasets are publicly available at `https://anonymous.4open.science/r/DeFence-57B0`.

2 Related Works

2.1 GNNs under Structure or Label Noise

Structural noise from spurious or missing edges disrupts message passing and degrades GNN predictions (Dong & Kluger, 2023). Existing defenses either purify or relearn the graph structure (Wu et al., 2019; Entezari et al., 2020; Jin et al., 2020), or reduce the influence of unreliable neighbors during propagation (Zhang & Zitnik, 2020; Geisler et al., 2021; Hou et al., 2024). Recent works extend these directions: Linkerhägner et al. (2025) jointly rewire the graph and denoise node features through spectral alignment, Zhuang et al. (2025) refine unreliable neighborhoods through contrastive learning, and Lee & Park (2025) recover graphs from adversarial structural perturbations. Liang et al. (2026) use attribute-guided prompts to compensate for representation shifts caused by structure noise.

Node annotations in real graphs are also often noisy. Existing methods mitigate label noise through reliable-sample selection, pseudo-label refinement, or label-agnostic regularization (Dai et al., 2021; Qian et al., 2023; Xia et al., 2021; Wu et al., 2024; Yuan et al., 2023; Xianxian Li, 2024). Recent works estimate label reliability from peer consistency (Li et al., 2025) or reconstruct topology-derived features to prevent overfitting to noisy labels (Wang et al., 2025), while probabilistic approaches model clean and noisy labels as latent variables (Qingqing Ge, 2023). However, these methods typically address only one noise source at a time: structure-noise defenses assume reliable supervision, whereas label-noise defenses depend on the observed topology.

2.2 GNNs under Joint Label-Structure Noise

When labels and edges are jointly corrupted, message passing couples their errors, yet this setting remains relatively underexplored. In et al. (2025) assume that feature noise causally induces structure and label noise, whereas we consider independently corrupted labels and edges without requiring feature noise. Chiu & Li (2026) address adversarial attacks, edge sparsity, and noisy labels rather than isolating the interaction between label and edge corruption. Huang et al. (2026) consider both corruptions, but specifically for binary fraud detection rather than general node classification.

RNCGLN (Zhu et al., 2024) and GDF (Zhang et al., 2026) match our setting. RNCGLN alternates between pseudo-labeling and graph refinement, whereas GDF uses separate node and edge networks

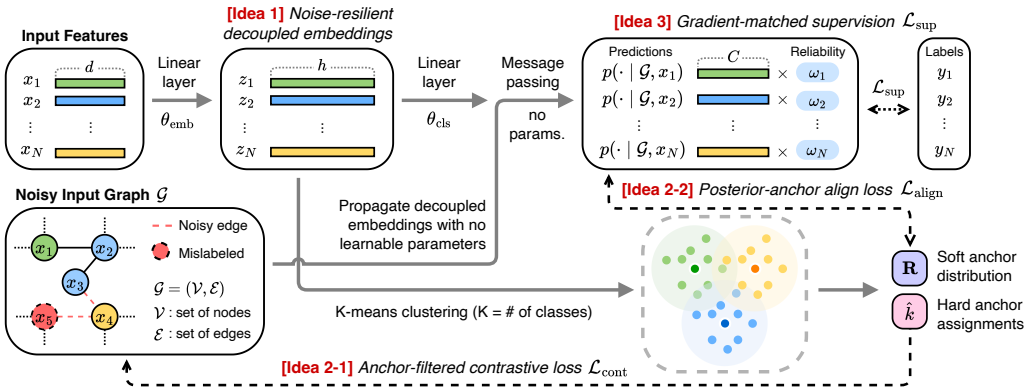

Figure 2: Overview of DEFENCE. *Idea 1:* cluster decoupled embeddings $\mathbf{Z}$ to obtain soft anchor memberships $\mathbf{R}$ and hard assignments $\hat{k}$. *Idea 2:* use anchors as a noise-resilient reference to robustify propagation and stabilize predictions: (Idea 2-1) anchor-filtered contrastive loss $\mathcal{L}_{\text{cont}}$ on observed edges to suppress spurious links, and (Idea 2-2) posterior-alignment loss $\mathcal{L}_{\text{align}}$ that pulls $p(\cdot \mid \mathcal{G}, x_i)$ toward the anchor distribution to stabilize predictions. *Idea 3:* train labeled nodes with gradient-matched sample weights ω_i to downweigh mislabeled instances.

to cross-correct their predictions. Despite their different designs, both obtain correction signals from models trained on the corrupted graph, leaving label and edge correction coupled. In contrast, DEFENCE constructs anchors from pre-propagation feature embeddings without directly using observed edges or labels. These anchors provide a separate reference for filtering unreliable structural interactions and regularizing noisy supervision.

3 PROPOSED METHOD

We propose DEFENCE, a robust node classification method under joint label-edge noise. Figure 2 and Algorithm 1 (Appendix A.1) illustrate the pipeline. We first state the core challenges and our ideas.

1. **How do we prevent noisy edges and noisy labels from negatively reinforcing each other?** We construct feature-derived anchors from pre-propagation embeddings as a noise-resilient reference. The refined embeddings are periodically reclustered to update the anchors, forming an iterative refinement process (Section 3.1).
2. **How do we use feature-derived anchors to refine node representations and predictions?** We use anchors for i) anchor-filtered contrastive learning that suppresses spurious edges and ii) anchor-posterior alignment that stabilizes post-propagation predictions (Section 3.2). We theoretically show that even weakly informative anchors provide effective regularization by yielding more class-consistent contrastive signals and selectively correcting predictions (Section 3.4).
3. **How do we reduce the influence of corrupted training labels?** We construct a reference gradient from confident predictions and downweigh labeled nodes whose gradients disagree with the reference (Section 3.3).

3.1 DECOUPLED ANCHOR CONSTRUCTION

In most existing robust GNNs, feature transformation and neighborhood aggregation are *coupled* within each layer (Jin et al., 2020). If the observed graph structure is corrupted, the first layer aggregates spurious neighbor signals and the transformation then learns from these distorted inputs. Subsequent layers carry this distortion forward and amplify it. When labels are also noisy, supervision fails to correct the drift and instead injects misleading gradients. As a result, the coupled design amplifies errors and degrades learning.

To break this coupling, we *decouple* feature embedding from propagation and supervision: the encoder first produces pre-propagation embeddings, which are then propagated for node prediction. A key idea is to use the pre-propagation embeddings to construct class-wise anchors and node-wise

soft memberships, which serve as a *noise-resilient* reference for regularizing noisy message passing and supervision. The anchors are periodically refreshed from the refined embeddings, forming an iterative refinement process. This design is architecture-agnostic and applies to any GNN with separate feature encoding and propagation modules. In our experiments, we instantiate this split with APPNP (Gasteiger et al., 2019).

Anchor construction. Given node features $\mathbf{X} = [x_1, ..., x_N]^\top \in \mathbb{R}^{N\times d}$ where N and d are the numbers of nodes and input features, respectively, we first build the feature-only embeddings by applying a linear layer followed by an activation function:

$$z_i = \sigma(\mathbf{W}_1 x_i + b_1), \qquad i = 1, ..., N, \tag{1}$$

where σ is ReLU, $\mathbf{W}_1 \in \mathbb{R}^{h\times d}$, $b_1 \in \mathbb{R}^h$, and h is the hidden embedding size. We denote the parameters $(\mathbf{W}_1, b_1)$ as θ_{emb} and stack z_i to form $\mathbf{Z} = [z_1, ..., z_N]^\top \in \mathbb{R}^{N\times h}$. We then cluster $\mathbf{Z}$ into C groups using K-Means (Lloyd, 1982), where C is the number of classes, and denote the resulting centroids by $\{\mu_c\}_{c=1}^C$. For each node, we define its soft membership over the anchors as

$$r_i(c) = \frac{\exp\big(-\|z_i - \mu_c\|_2^2\big)}{\sum_{c'=1}^C \exp\big(-\|z_i - \mu_{c'}\|_2^2\big)}, \quad i = 1, ..., N,\ c = 1, ..., C. \tag{2}$$

We refer to $r_i \in \mathbb{R}^C$ as the *anchor distribution* and stack them to form anchor distribution matrix $\mathbf{R} = [r_1, ..., r_N]^\top \in \mathbb{R}^{N\times C}$.

Since K-means assigns cluster IDs arbitrarily, we align them with the class indices using the training split. Specifically, we form the cluster-class confusion matrix $\mathbf{M} \in \mathbb{N}^{C\times C}$ where $\mathbf{M}(c, y)$ counts how many training nodes in cluster c have class y. We then choose the one-to-one mapping that reorders the columns of $\mathbf{M}$ so that the sum of its diagonal entries, which represents the total number of cluster-class agreements, is maximized. This linear assignment is solved by the Hungarian algorithm (Kuhn, 1955), yielding a permutation matrix $\Pi \in \{0, 1\}^{C\times C}$. With this, we form the class-aligned anchor distribution $\tilde{r}_i = \Pi^\top r_i$ and hard anchor assignments $\hat{k}_i = \arg\max_c \tilde{r}_i(c)$. For notational convenience, we will drop the tilde and simply use r_i to denote $\tilde{r}_i$.

Anchor refresh and prediction. We refresh $\mathbf{R}$ and $\hat{k}$ intermittently every T epochs. We set $T = 10$ in all of our experiments. Within each refresh window, $\mathbf{R}$ and $\hat{k}$ are treated as constant. This prevents message passing and labels from directly leaking into the anchors, preserving their role as feature-derived references. After obtaining $\mathbf{R}$ and $\hat{k}$, we apply a linear transformation $\theta_{\text{cls}} = (\mathbf{W}_2, b_2)$ on $\mathbf{Z}$ to match the embedding dimension to the number of target classes ($\mathbf{W}_2 \in \mathbb{R}^{C\times h}, b_2 \in \mathbb{R}^C$). The resulting logits are then fed into the propagation stage.

In the following Section 3.2, we describe how $\mathbf{R}$ and $\hat{k}$ are used to regularize the joint noise.

3.2 Anchor-Based Regularization under Joint Noise

Under joint noise, spurious edges distort the embedding space by pulling unrelated nodes together during message passing, while corrupted labels miscalibrate posteriors through supervision. To address this, we propose two anchor-guided components: i) an *anchor-filtered contrastive loss* that forms positive pairs only along edges consistent with the anchor assignments, suppressing the impact of spurious links, and ii) *anchor-posterior alignment loss* that tethers predictions to the anchor distribution, reducing drift induced by corrupted labels and noisy propagation.

Anchor-filtered contrastive loss. The goal is to refine the feature-only embeddings $\mathbf{Z}$ so that subsequent message passing becomes less sensitive to spurious edges. The anchor-filtered contrastive loss $\mathcal{L}_{\text{cont}}$ defines positive pairs in $\mathbf{Z}$ as observed edges whose endpoints are anchor-consistent, and refines $\mathbf{Z}$ by pulling such pairs together while pushing cross-anchor pairs apart. This encourages $\mathbf{Z}$ to encode anchor-consistent graph structure, instead of indiscriminately using all observed edges in the noisy graph. As a result, pre-propagation embeddings become more separable across anchors, making post-propagation predictions less affected by spurious edges.

Using hard anchor assignments $\hat{k}_i$, we form candidate sets $\mathcal{C}_+$ and $\mathcal{C}_-$ of positive and negative pairs, respectively: $\mathcal{C}_+ = \{(i, j) \in \mathcal{E} : \hat{k}_i = \hat{k}_j\}$ and $\mathcal{C}_- = \{(i, j) \in [N] \times [N] : \hat{k}_i \neq \hat{k}_j\}$ where $\mathcal{E}$

is the set of observed edges in the given graph $\mathcal{G}$ and $[N] \times [N]$ denotes all the node pairs. From these we draw sets $\mathcal{S}_+$ and $\mathcal{S}_-$ of positive and negative node pairs of equal size B from $\mathcal{C}_+$ and $\mathcal{C}_-$, respectively. Let z_i be the pre-propagation embedding from Equation (1) and $\tilde{z}_i = z_i/\|z_i\|_2$. Then we define $\mathcal{L}_{\text{cont}}$ as follows:

$$\mathcal{L}_{\text{cont}} = \frac{1}{2B}\left(\sum\nolimits_{(i,j)\in\mathcal{S}_+}\left(1-\tilde{z}_i^\top\tilde{z}_j\right)+\sum\nolimits_{(i,j)\in\mathcal{S}_-}\max\left(0,\tilde{z}_i^\top\tilde{z}_j-m\right)\right) \tag{3}$$

where m is a constant margin for negative samples, which acts as a safeguard that suppresses the effect of ambiguous false negatives. We set $B = 1024$ and $m = 0.5$ in all of our experiments.

Importantly, $\mathcal{L}_{\text{cont}}$ does not require highly accurate anchors. As shown in Theorem 1, even weakly informative anchors—i.e., anchors that are only slightly better than random—yield cleaner pairwise supervision than the observed graph structure alone. Thus, even imperfect anchors allow $\mathcal{L}_{\text{cont}}$ to provide a more reliable regularizer than conventional structure-based contrastive objectives.

Posterior-anchor alignment loss. The goal is to regularize the posterior toward a feature-derived anchor reference to prevent drift from corrupted labels and noisy propagation. Concretely, the alignment loss $\mathcal{L}_{\text{align}}$ tethers the model posterior to the soft anchor distribution $\mathbf{R}$.

Let $o_i \in \mathbb{R}^C$ be the post-propagation logits of node i, and $p_\theta(\cdot \mid \mathcal{G}, x_i) = \text{softmax}(o_i)$ where θ consists of two linear layers θ_{emb} and θ_{cls}. We denote $p_\theta(\cdot \mid \mathcal{G}, x_i)$ as $p_\theta(\cdot \mid i)$ for brevity. We propose an alignment loss $\mathcal{L}_{\text{align}}$ that keeps post-propagation predictions $p_\theta(\cdot \mid i)$ consistent with the feature-derived anchor distribution r_i:

$$\mathcal{L}_{\text{align}} = \frac{1}{|\mathcal{D}_{\text{trn}}|}\sum_{i\in\mathcal{D}_{\text{trn}}}\text{KL}\left(r_i \,\|\, p_\theta(\cdot \mid i)\right) = \frac{1}{|\mathcal{D}_{\text{trn}}|}\sum_{i\in\mathcal{D}_{\text{trn}}}\sum_{c=1}^{C} r_i(c)\big(\log r_i(c) - \log p_\theta(c \mid i)\big). \tag{4}$$

Importantly, even under a mild assumption that the anchors are only slightly better than random guessing, $\mathcal{L}_{\text{align}}$ remains a selective regularizer. As shown in Theorem 2, $\mathcal{L}_{\text{align}}$ induces a stronger output-level correction when predictions follow incorrect observed labels than when predictions follow clean labels.

3.3 Gradient-matched Supervision

While Section 3.2 improves prediction robustness using anchors as a noise-resilient reference, supervised training still relies on corrupted labels, making per-example reliability important. Existing approaches often use model confidence on training nodes as a reliability signal for identifying noisy labels. However, these predictions are learned from the same noisy supervision, leading to self-reinforcement of label errors. Instead, we construct a reference gradient from confident predictions on non-training nodes and weigh each labeled training node by its gradient agreement with this reference. This separates reliability estimation from the noisy training labels and reduces their influence during supervision. To compute the agreement efficiently, we use only the class-matching layer θ_{cls}, i.e., the last learnable layer before propagation, whose gradients directly reflect the label supervision signal.

For each labeled training node $i \in \mathcal{D}_{\text{trn}}$, let ℓ_i denote its supervised cross-entropy loss: $\ell_i = -\sum_{c=1}^{C} y_i(c)\log p_\theta(c \mid i)$ with one-hot label y_i and its c-th component $y_i(c)$. Among the non-training nodes, we select the top 10% with the highest prediction confidence and use their predicted classes as pseudo-labels. Let $\mathcal{D}_{\text{ref}}$ denote this reference set and $\tilde{y}_j = \arg\max_{c\in[C]} p_\theta(c \mid j)$ the pseudo-label of node j. The reference gradient in terms of θ_{cls} is computed as $\frac{\partial \ell_{\text{ref}}}{\partial \theta_{\text{cls}}}$ where $\ell_{\text{ref}} = -\frac{1}{|\mathcal{D}_{\text{ref}}|}\sum_{j\in\mathcal{D}_{\text{ref}}}\log p_\theta(\tilde{y}_j \mid j)$. Then the degree α_i of gradient alignment of node i with the reference gradient is computed by: $\alpha_i = \langle \frac{\partial \ell_i}{\partial \theta_{\text{cls}}}, \frac{\partial \ell_{\text{ref}}}{\partial \theta_{\text{cls}}} \rangle$ where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product over the classifier-layer parameter blocks.

α_i measures how well node i's update aligns with the reference direction. We convert these scores into per-node reliability weights ω_i that modulate supervision. This mapping should be i) monotone, to preserve the ordering by agreement, ii) bounded, so that a few large scores do not dominate optimization and training remains stable, and iii) strictly positive, so uncertain points retain some influence. Accordingly, we normalize α_i over $\mathcal{D}_{\text{trn}}$ and rescale it to $[0.5, 1.0]$:

$$\omega_i = 0.5\left(1 + \frac{\max(\alpha_i, 0)}{\max(\{\alpha_i\}_{i\in\mathcal{D}_{\text{trn}}})}\right). \tag{5}$$

Then the reliability-weighted supervised loss $\mathcal{L}_{\text{sup}}$ is defined by a weighted cross-entropy as follows:

$$\mathcal{L}_{\text{sup}} = -\frac{1}{\sum_{i\in\mathcal{D}_{\text{trn}}}\omega_i}\sum\nolimits_{i\in\mathcal{D}_{\text{trn}}}\omega_i\sum\nolimits_{c=1}^{C} y_i(c)\,\log p_\theta(c \mid i) \tag{6}$$

where $\sum_i \omega_i$ keeps the loss scale stable across epochs.

Overall objective. Combining Equations (3), (4), and (6), the final objective $\mathcal{L}(\theta)$ is

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{sup}} + \lambda_{\text{cont}}\mathcal{L}_{\text{cont}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}}, \tag{7}$$

where λ_{cont} and λ_{align} are balancing parameters and $\theta = \{\theta_{\text{emb}}, \theta_{\text{cls}}\}$ denotes the learnable model parameters. Here, $\mathcal{L}_{\text{sup}}$ fits the observed labels with gradient-based weights, $\mathcal{L}_{\text{cont}}$ applies anchor-filtered pairwise regularization, and $\mathcal{L}_{\text{align}}$ limits propagation-induced posterior drift. The anchors enter the objective only through $\mathcal{L}_{\text{cont}}$ and $\mathcal{L}_{\text{align}}$ as regularization signals.

3.4 Theoretical Analysis

3.4.1 Reliability and Selectivity of Anchor-Based Losses

Analysis on anchor-filtered contrastive loss. Existing structure-based contrastive learning treats neighboring nodes as positive pairs and other sampled nodes as negative pairs (Hu et al., 2021; Zhu et al., 2024). However, these pools contain cross-class positive pairs and same-class negative pairs under noise graphs, resulting in incorrect pull and push signals. Our anchor filter adds a second criterion to these pair pools: positive edges are retained when their endpoints share an anchor, while negative pairs are retained when their anchors differ. We analyze the resulting change in class consistency relative to unfiltered pair sampling.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the observed graph with $N = |\mathcal{V}|$ nodes and C classes, with $[N] = 1, \ldots, N$ and $[C] = 1, \ldots, C$. For each node $i \in \mathcal{V}$, let $y_i^\star \in [C]$ denote its ground-truth class and $\hat{k}_i \in [C]$ its hard anchor assignment. For $u, v \in \mathcal{V}$, define $T_{uv} = \mathbf{1}\{y_u^\star = y_v^\star\}$ and $A_{uv} = \mathbf{1}\{\hat{k}_u = \hat{k}_v\}$ where T_{uv} and A_{uv} indicate clean-class agreement and anchor agreement, respectively.

Without anchor filtering, the positive pool reduces to the observed edge set $\mathcal{E}$, while the negative pool is the set of distinct node pairs $\mathcal{P} = \{(u, v) \in [N]\times[N] : u \neq v\}$. Sampling (i, j) uniformly from $\mathcal{E}$ and (u, v) uniformly from $\mathcal{P}$, we define $h = P(T_{ij} = 1)$, $\tau = P(T_{uv} = 0)$, $p_+ = P(T_{ij} = 1 \mid A_{ij} = 1)$, and $p_- = P(T_{uv} = 0 \mid A_{uv} = 0)$. Here, h and τ quantify class consistency without anchor filtering, whereas p_+ and p_- quantify it after filtering into $\mathcal{C}_+$ and $\mathcal{C}_-$, respectively.

We assume that the class-aligned hard anchor assignments are more accurate than random guessing. Specifically, we consider a symmetric anchor error model with $\epsilon_{\text{anc}} \in [0, (C-1)/C)$, where ϵ_{anc} denotes the probability of assigning a node to an incorrect anchor:

$$P(\hat{k}_i = c \mid y_i^\star = y) = \begin{cases} 1 - \epsilon_{\text{anc}}, & c = y, \\ \epsilon_{\text{anc}}/(C-1), & c \neq y. \end{cases} \tag{8}$$

The upper bound $\epsilon_{\text{anc}} < (C-1)/C$ is exactly the better-than-random condition, since it ensures $P(\hat{k}_i = y \mid y_i^\star = y) = 1 - \epsilon_{\text{anc}} > 1/C$.

Theorem 1 (Reliability gain by anchor filtering)**.** *Under the anchor error assumption in Equation (8). If $0 < h < 1$ and $0 < \tau < 1$, anchor filtering increases the clean-class reliability of the positive and negative candidate pools by at least*

$$p_+ - h \geq h(1-h)\Delta_{\text{anc}}, \qquad p_- - \tau \geq \tau(1-\tau)\Delta_{\text{anc}}. \tag{9}$$

where $\Delta_{\text{anc}} = \left(1 - C\epsilon_{\text{anc}}/(C-1)\right)^2$. The proof is provided in Appendix A.3.1.

Analysis on posterior-anchor alignment. We analyze how $\mathcal{L}_{\text{align}}$ acts as a selective regularizer by imposing weak regularization on clean nodes and strong regularization on mislabeled nodes.

We assume that the anchors are better than random guessing: $\mathbb{E}[r_i(y) \mid y_i^\star = y] > 1/C$ for every $y \in [C]$ where r_i is the anchor distribution. For a node i with $y_i^\star = y$, we consider two cases: its observed label is y (clean) or an incorrect class $c \neq y$ (mislabeled). To compare the two cases at the

same prediction confidence, we assume that the posterior assigns the same probability $s \in (1/C, 1)$ to the observed label and $\bar{s} = (1-s)/(C-1)$ to each remaining class. We denote the resulting posteriors by $p_\theta^{\text{clean}}(\cdot \mid i)$ and $p_\theta^{\text{noisy},c}(\cdot \mid i)$, respectively. Thus, the two posteriors assign the same probability s to their respective observed labels and differ only in whether the observed label is correct.

Theorem 2 (Selective regularization by anchor alignment)**.** *Under the above assumptions, the alignment term produces a larger squared gradient with respect to the post-propagation logits o_i for $p_\theta^{\text{noisy},c}$ than for p_θ^{clean}, on average over nodes of class y and all incorrect classes $c \neq y$:*

$$\frac{1}{C-1}\sum_{c\neq y}\mathbb{E}\left[\left\|\nabla_{o_i}\,\mathrm{KL}\Big(r_i \,\|\, p_\theta^{\text{noisy},c}(\cdot \mid i)\Big)\right\|_2^2 \mid y_i^\star = y\right] > \mathbb{E}\left[\left\|\nabla_{o_i}\,\mathrm{KL}\Big(r_i \,\|\, p_\theta^{\text{clean}}(\cdot \mid i)\Big)\right\|_2^2 \mid y_i^\star = y\right]. \tag{10}$$

Thus, $\mathcal{L}_{\text{align}}$ applies a stronger output-level correction when the posterior follows an incorrect label and a weaker correction when it follows the clean label. The proof is provided in Appendix A.3.2.

3.4.2 Scalability

We analyze the scalability of DeFence and show that its overall training time scales linearly with the number of nodes and edges. For simplicity, we analyze the worst-case where anchors are refreshed every epoch; in practice, we refresh anchors every T epochs (Section 3.1), which reduces the K-means overhead by a factor of T.

Theorem 3. *We are given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ and $\mathcal{E}$ are sets of nodes and edges, respectively. Let d, h, and C be the input feature dimension, the hidden dimension, and the number of classes, respectively. Then the time complexity of* DeFence *for each epoch is*

$$\mathcal{O}\left((dh + I_k Ch + I_j)|\mathcal{V}| + I_j|\mathcal{E}|\right) \tag{11}$$

which is linear in the number of nodes and edges. (see Appendix A.3.3 for proof).

4 Experiments

Through experiments, we answer the following questions.

- **Q1. Classification performance (Section 4.2).** Does DeFence outperform robust baselines under the joint label-structure noise?
- **Q2. Visualization (Section 4.3).** Do the anchors of DeFence learn class-consistent representations and reliably guide robust learning?
- **Q3. Ablation study (Section 4.4).** How much do the three components—feature-only anchors, posterior-anchor alignment, and gradient-matched supervision—each contribute?
- **Q4. Hyperparameter sensitivity (Section 4.5).** How sensitive is DeFence to λ_{cont} and λ_{align}?
- **Q5. Scalability (Section 4.6).** How does DeFence's runtime scale with graph size?

4.1 Experimental Settings

Setup. We evaluate on four standard node-classification benchmarks: Cora, PubMed, Citeseer-full, and Flickr, listed in Table 4 (Appendix A.2). We evaluate under joint label-structure noise. At label-noise rate ρ_l, we randomly flip the labels of ρ_l of the labeled training nodes to incorrect classes; at structure-noise rate ρ_s, we add $\rho_s|\mathcal{E}|$ uniformly sampled non-edges. Appendices A.6 and A.7 additionally evaluate pairwise label noise and random edge removal, respectively. We report node classification accuracy over five random seeds using a single NVIDIA GTX 1080 Ti GPU.

Baselines. We compare DeFence against four families of methods. GCN (Kipf & Welling, 2017) is a canonical GNN baseline with robust performance. For label-noise methods, we include NRGNN (Dai et al., 2021), TFR (Wang et al., 2025), and ProCon (Li et al., 2025). For structure-noise methods, we include GCN-SVD (Entezari et al., 2020), GNNGuard (Zhang & Zitnik, 2020), RGCN (Zhu et al., 2019), and NoisyGNN (Dai et al., 2022). For joint label-structure noise, we include RNCGLN (Zhu et al., 2024) and GDF (Zhang et al., 2026), the only methods among our baselines designed to handle both noises simultaneously. We use the hyperparameter settings reported in the original papers unless otherwise specified in Appendix A.4.

Table 1: The node classification accuracy of DEFENCE and baselines under different ratios of label noise (ρ_l) and structure noise (ρ_s). Best results are in bold and second-best are underlined. O.O.M. denotes the out-of-memory error. Note that DEFENCE outperforms the baselines in most settings.

Datasets	$\{\rho_l, \rho_s\}$	GCN	NRGNN	TFR	ProCon	GCN-SVD	GNNGuard	RGCN	NoisyGNN	RNCGLN	GDF	DEFENCE (ours)
Cora	{0.1, 0.1}	72.2 ± 1.1	64.8 ± 7.4	72.9 ± 2.0	55.4 ± 0.8	56.1 ± 0.1	64.3 ± 0.1	73.4 ± 0.0	72.3 ± 0.0	73.3 ± 1.4	67.9 ± 0.2	**79.0 ± 2.2**
	{0.1, 0.2}	70.2 ± 1.0	64.1 ± 7.3	66.2 ± 0.9	51.2 ± 2.2	55.7 ± 0.1	63.9 ± 0.0	70.3 ± 0.0	69.3 ± 0.0	72.1 ± 1.7	64.3 ± 1.9	**75.4 ± 1.2**
	{0.2, 0.1}	67.1 ± 2.1	61.0 ± 7.8	67.1 ± 0.6	52.5 ± 3.3	54.9 ± 0.1	58.1 ± 0.1	68.6 ± 0.1	67.8 ± 0.0	72.7 ± 1.9	62.0 ± 0.4	**73.7 ± 0.1**
	{0.2, 0.2}	64.7 ± 2.0	56.7 ± 8.5	65.1 ± 0.8	52.0 ± 6.0	51.6 ± 0.2	57.5 ± 0.1	64.9 ± 0.1	64.6 ± 0.1	72.4 ± 1.1	58.8 ± 2.9	**72.6 ± 1.4**
PubMed	{0.1, 0.1}	70.4 ± 2.0	69.4 ± 1.8	70.2 ± 2.9	64.6 ± 2.1	65.7 ± 0.2	70.6 ± 0.1	69.8 ± 0.0	70.4 ± 0.0	O.O.M.	69.4 ± 3.3	**75.0 ± 2.4**
	{0.1, 0.2}	68.5 ± 1.7	66.9 ± 2.1	69.3 ± 1.1	62.1 ± 2.4	66.4 ± 0.1	70.7 ± 0.1	67.0 ± 0.2	69.8 ± 0.0	O.O.M.	67.4 ± 0.8	**73.9 ± 1.5**
	{0.2, 0.1}	65.3 ± 3.2	66.1 ± 3.6	68.5 ± 2.2	62.9 ± 1.6	62.5 ± 0.1	64.9 ± 0.1	66.5 ± 0.1	70.9 ± 0.0	O.O.M.	64.1 ± 2.1	**71.7 ± 0.7**
	{0.2, 0.2}	64.5 ± 2.6	63.4 ± 3.9	66.4 ± 2.0	62.0 ± 2.7	63.0 ± 0.1	64.5 ± 0.1	64.1 ± 0.0	66.0 ± 0.2	O.O.M.	63.3 ± 3.1	**69.9 ± 0.7**
Citeseer	{0.1, 0.1}	81.8 ± 0.4	78.6 ± 4.5	82.7 ± 0.8	67.1 ± 4.6	72.2 ± 0.6	72.7 ± 0.7	82.4 ± 0.5	82.1 ± 1.2	77.7 ± 0.7	75.1 ± 0.4	**83.1 ± 1.8**
	{0.1, 0.2}	79.7 ± 1.6	76.4 ± 5.9	**79.9 ± 1.3**	66.1 ± 2.3	70.6 ± 1.1	71.9 ± 1.3	79.5 ± 0.7	78.8 ± 1.0	75.1 ± 2.1	72.1 ± 0.7	79.7 ± 1.0
	{0.2, 0.1}	79.5 ± 1.0	78.1 ± 1.6	**82.3 ± 0.7**	65.4 ± 4.0	68.5 ± 1.5	70.6 ± 1.3	78.7 ± 1.0	79.0 ± 0.8	76.8 ± 0.8	67.7 ± 0.9	81.7 ± 1.1
	{0.2, 0.2}	77.1 ± 1.3	75.7 ± 1.8	78.2 ± 0.7	62.8 ± 4.9	68.1 ± 1.9	70.4 ± 1.3	75.8 ± 1.0	75.9 ± 2.0	75.2 ± 2.4	64.8 ± 1.4	**78.8 ± 1.8**
Flickr	{0.1, 0.1}	55.5 ± 1.3	45.7 ± 3.4	24.9 ± 2.2	38.3 ± 1.1	63.9 ± 1.9	67.8 ± 3.9	28.5 ± 1.8	54.0 ± 0.5	68.9 ± 1.7	54.6 ± 0.3	**75.1 ± 0.5**
	{0.1, 0.2}	53.2 ± 1.0	44.3 ± 2.4	22.1 ± 1.7	35.6 ± 1.2	63.3 ± 0.7	67.5 ± 3.2	24.6 ± 2.0	54.6 ± 1.1	64.0 ± 1.9	54.9 ± 0.7	**74.3 ± 1.9**
	{0.2, 0.1}	53.0 ± 2.4	39.8 ± 3.8	26.3 ± 1.0	37.8 ± 1.2	61.5 ± 0.3	61.7 ± 3.5	27.9 ± 3.0	52.4 ± 1.2	68.7 ± 1.6	51.0 ± 0.8	**71.5 ± 1.4**
	{0.2, 0.2}	50.5 ± 3.0	38.5 ± 2.9	21.0 ± 0.7	32.7 ± 2.2	61.5 ± 1.2	61.0 ± 3.0	19.0 ± 5.3	52.2 ± 0.5	63.9 ± 1.6	51.3 ± 0.2	**69.0 ± 2.3**

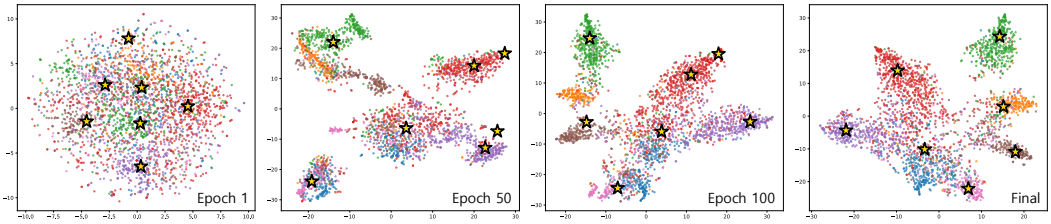

Figure 3: Evolution of feature-only embeddings $\mathbf{Z}$ on the Cora dataset, visualized using t-SNE. Points denote node embeddings colored by ground-truth labels, and stars indicate K-means centroids obtained without label information. As training progresses, embeddings form clear class-wise clusters, and the centroids emerge near their centers. They act as effective class representatives, improving the anchor distribution $\mathbf{R}$ over time.

4.2 PERFORMANCE

We evaluate DEFENCE under joint noise by sweeping the label-noise rate ρ_l and the structure-noise rate ρ_s over $\{0.1, 0.2\}$ and reporting all combinations. The results are summarized in Table 1. Additional results under pair-wise label noise, missing-edge structure noise, and extreme noise ratios are provided in Appendices A.6, A.7, and A.5, respectively.

As the noise intensifies, accuracy generally degrades. Specifically, increases in ρ_l typically cause a larger drop than equal increases in ρ_s, since label flips directly corrupt the supervision signal, whereas structural perturbations affect aggregation and are partly smoothed during propagation.

Single-noise baselines degrade substantially under joint corruption, highlighting the limitations of addressing either noise alone. RNCGLN, which explicitly models both noise sources, is typically the runner-up on Citeseer, Cora, and Flickr, but on PubMed it encounters out-of-memory failures. Overall, DEFENCE achieves the best node classification accuracy in most settings.

4.3 VISUALIZATION

Figure 3 illustrates the evolution of feature-only embeddings $\mathbf{Z}$ during training, visualized using t-SNE. Each point denotes a node embedding colored by its ground-truth label, while stars represent the centroids obtained by K-means without access to labels.

At the early stage (epoch 1), embeddings from different classes are highly entangled. As training proceeds (epochs 50, 100, and final), embeddings become more compact within classes and well separated across classes. Remarkably, the K-means centroids—computed in a completely unsupervised manner—emerge near the centers of the ground-truth class clusters, thereby serving as reliable class representatives. Consequently, the anchor distribution $\mathbf{R}$ derived from these centroids becomes progressively more accurate, enabling the anchor-based contrastive and alignment losses to operate more effectively.

4.4 Ablation Study

We evaluate the contribution of each component by removing it individually under label- and structure-noise rates $\rho_l, \rho_s \in \{0.1, 0.2\}$. *w/o-*$\mathcal{L}_{\text{cont}}$ removes the anchor-filtered contrastive regularizer, while *w/o-*$\mathcal{L}_{\text{align}}$ removes the anchor-posterior alignment. *w/o-grad* replaces gradient-matched supervision with unweighted cross-entropy.

Table 2: Ablation study on DEFENCE.

Data	$\{\rho_l, \rho_s\}$	w/o-$\mathcal{L}_{\text{cont}}$	w/o-$\mathcal{L}_{\text{align}}$	w/o-grad	DEFENCE
Cora	$\{0.1, 0.1\}$	77.5 ± 2.6	77.7 ± 1.1	78.3 ± 0.2	**79.0 ± 2.2**
	$\{0.1, 0.2\}$	73.8 ± 2.9	74.1 ± 2.7	75.7 ± 1.7	**75.4 ± 1.2**
	$\{0.2, 0.1\}$	72.3 ± 1.1	72.0 ± 0.8	73.3 ± 2.6	**73.7 ± 0.1**
	$\{0.2, 0.2\}$	67.6 ± 2.0	71.1 ± 1.8	59.9 ± 24.3	**72.6 ± 1.4**
PubMed	$\{0.1, 0.1\}$	74.8 ± 1.5	74.4 ± 3.8	74.8 ± 1.2	**75.0 ± 2.4**
	$\{0.1, 0.2\}$	73.5 ± 1.9	73.8 ± 1.5	73.8 ± 1.4	**73.9 ± 1.5**
	$\{0.2, 0.1\}$	70.1 ± 0.4	71.6 ± 0.9	71.3 ± 2.0	**71.7 ± 0.7**
	$\{0.2, 0.2\}$	68.7 ± 2.2	68.8 ± 2.2	69.6 ± 1.5	**69.9 ± 0.7**
Flickr	$\{0.1, 0.1\}$	75.1 ± 0.5	75.1 ± 0.5	74.7 ± 1.1	**75.1 ± 0.5**
	$\{0.1, 0.2\}$	73.4 ± 2.3	74.0 ± 1.3	74.2 ± 1.3	**74.3 ± 1.9**
	$\{0.2, 0.1\}$	69.7 ± 1.9	69.9 ± 1.5	69.9 ± 1.9	**71.5 ± 1.4**
	$\{0.2, 0.2\}$	68.7 ± 2.2	68.4 ± 1.2	68.7 ± 1.4	**69.0 ± 2.3**

Removing any component consistently reduces the mean accuracy, demonstrating that all three components contribute to robustness under joint label-structure noise.

4.5 Hyperparameter Sensitivity

We evaluate sensitivity to λ_{cont} and λ_{align} under fixed label and structure noise ($\rho_l = 0.1,\ \rho_s = 0.1$) on Citeseer (Figure 4, left) and PubMed (Figure 4, right).

Figure 4 shows comparable accuracy across a broad range of hyperparameter combinations, indicating that DEFENCE does not require aggressive tuning.

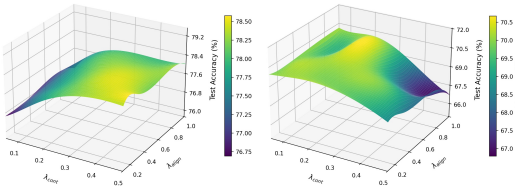

Figure 4: Sensitivity to λ_{cont} and λ_{align}. Accuracy remains stable over a broad range.

4.6 Scalability

We empirically validate Theorem 3 by measuring wall-clock time under controlled graph sizes. Starting from the original graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we randomly remove a fraction $\rho_v \in \{0.1, 0.2, \ldots, 0.5\}$ of nodes and take the induced subgraph on the remaining nodes, with node retain ratio $r_v = 1 - \rho_v$. For each setting, we report runtime over 100 training epochs.

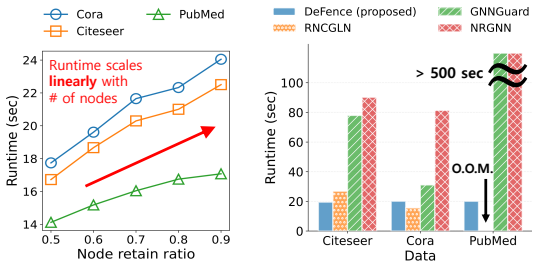

Figure 5: Runtime of DEFENCE with varying node retain ratios (left) and comparison with baselines (right).

As shown in Figure 5 (left), runtime grows approximately linearly with the node retain ratio. PubMed exhibits the shortest runtime because DEFENCE computes gradient-alignment scores in the label space, whose cost scales with the number of classes, and PubMed has only three classes.

Figure 5 (right) compares the wall-clock runtime with recent baselines. DEFENCE remains consistently efficient, whereas GNNGuard and NRGNN are slower because they repeatedly compute edge reliability over all edges.

5 Conclusion

We study node classification under joint label-structure noise, a common yet underexplored setting where spurious edges and corrupted labels interact and are compounded. We propose DEFENCE, a robust classifier for this setting. DEFENCE first derives anchors from pre-propagation embeddings and periodically refreshes them. Between refreshes, these anchors remain fixed, providing a stable reference that is not directly affected by noisy message passing. During training, we use these anchors to regularize both structure and labels. Anchor-filtered contrastive learning filters graph-based pairwise constraints, posterior-anchor alignment limits propagation-induced prediction drift, and gradient-matched reweighting reduces the influence of unreliable labeled nodes. Across various real-world datasets, DEFENCE consistently achieves the best performance among strong baselines.

REFERENCES

Jun-Wei Chiu and Cheng-Te Li. HRGNN: Learning holistically robust graph neural networks on noisy graphs with label scarcity. *ACM Transactions on Intelligent Systems and Technology*, 2026.

Enyan Dai, Charu Aggarwal, and Suhang Wang. Nrgnn: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021.

Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, 2022.

Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. Credit card fraud detection: a realistic modeling and a novel learning strategy. *IEEE transactions on neural networks and learning systems*, 2018.

Mingze Dong and Yuval Kluger. Towards understanding and reducing graph structural noise for gnns. In *International Conference on Machine Learning*, 2023.

Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th international conference on web search and data mining*, 2020.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *International Conference on Learning Representations*, 2019.

Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems*, 2021.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 2017.

Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.

Zhichao Hou, Ruiqi Feng, Tyler Derr, and Xiaorui Liu. Robust graph neural networks via unbiased aggregation. *Advances in Neural Information Processing Systems*, 2024.

Yang Hu, Haoxuan You, Zhecan Wang, Zhicheng Wang, Erjin Zhou, and Yue Gao. Graph-MLP: Node classification without message passing in graph. *arXiv preprint arXiv:2106.04051*, 2021.

Yuting Huang, Ziquan Fang, Zhengjie Zhou, Tinghui Luo, Lu Chen, Surun Ji, Huimei Zheng, Mingfan Lu, Fangshu Chen, and Yunjun Gao. iQ-Guard: An effective and noise-resistant framework for graph fraud detection on iQIYI platform. In *Proceedings of the ACM Web Conference*, 2026.

Yeonjun In, Kanghoon Yoon, Sukwon Yun, Kibum Kim, Sungchul Kim, and Chanyoung Park. Training robust graph neural networks by modeling noise dependencies. In *Advances in Neural Information Processing Systems*, 2025.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955.

Woohyun Lee and Hogun Park. Self-supervised adversarial purification for graph neural networks. In *International Conference on Machine Learning*, 2025.

Kailai Li, Jiawei Sun, Jiong Lou, Zhanbo Feng, Hefeng Zhou, Chentao Wu, Guangtao Xue, Wei Zhao, and Jie Li. Leveraging peer-informed label consistency for robust graph neural networks with noisy labels. In *IJCAI*, 2025.

Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020.

Yayong Li, Jie Yin, and Ling Chen. Unified robust training for graph neural networks against label noise. In *Pacific-Asia conference on knowledge discovery and data mining*, 2021.

Zhuomin Liang, Liang Bai, and Xian Yang. Attribute-guided dynamic prompt learning for graph neural networks. In *AAAI Conference on Artificial Intelligence*, 2026.

Jonas Linkerhägner, Cheng Shi, and Ivan Dokmanić. Joint graph rewiring and feature denoising via spectral resonance. In *International Conference on Learning Representations*, 2025.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 1982.

Hoang NT, Choong Jun Jin, and Tsuyoshi Murata. Learning graph neural networks with noisy labels. 2019.

Siyi Qian, Haochao Ying, Renjun Hu, Jingbo Zhou, Jintai Chen, Danny Z Chen, and Jian Wu. Robust training of graph neural networks via noise governance. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, 2023.

Zeyuan Zhao Xiang Li Qingqing Ge, Jianxiang Yu. Resist label noise with pgm for graph neural networks. In *arXiv:2311.02116*, 2023.

Bharat Runwal, Vivek, and Sandeep Kumar. Robust graph neural networks using weighted graph laplacian. *arXiv preprint arXiv:2208.01853*, 2022.

Romy Sauvayre. Types of errors hiding in google scholar data. *Journal of Medical Internet Research*, 2022.

Zhonghao Wang, Yuanchen Bei, Sheng Zhou, Zhiyao Zhou, Jiapei Fan, Hui Xue, Haishuai Wang, and Jiajun Bu. Learning from graph: Mitigating label noise on graph through topological feature reconstruction. In *CIKM*, 2025.

Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: deep insights into attack and defense. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, 2019.

Yuhao Wu, Jiangchao Yao, Xiaobo Xia, Jun Yu, Ruxin Wang, Bo Han, and Tongliang Liu. Mitigating label noise on graphs via topological sample selection. In *International Conference on Machine Learning*, 2024.

Jun Xia, Haitao Lin, Yongjie Xu, Lirong Wu, Zhangyang Gao, Siyuan Li, and Stan Z. Li. Towards robust graph neural networks against label noise. ICLR 2021 OpenReview submission, 2021. URL `https://openreview.net/forum?id=H38f_9b90BO`.

De Li Haodong Qian Jinyan Wang Xianxian Li, Qiyu Li. Contrastive learning of graphs under label noise. *Neural networks*, 2024.

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

Yang Ye and Shihao Ji. Sparse graph attention networks. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

Jingyang Yuan, Xiao Luo, Yifang Qin, Yusheng Zhao, Wei Ju, and Ming Zhang. Learning on graphs under label noise. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

Mengmei Zhang, Linmei Hu, Chuan Shi, and Xiao Wang. Adversarial label-flipping attack and defense for graph neural networks. In *IEEE International Conference on Data Mining (ICDM)*, 2020.

Xiang Zhang and Marinka Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in neural information processing systems*, 2020.

Xiaobo Zhang, Yutao Liu, Hao Wang, Wei Xie, and Wei Wang. GNN Defender: Overcoming mixed noise through cross-correction over nodes and edges. *IEEE Transactions on Knowledge and Data Engineering*, 2026.

Jia-Xing Zhong, Nannan Li, Weijie Kong, Shan Liu, Thomas H Li, and Ge Li. Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019.

Yonghua Zhu, Lei Feng, Zhenyun Deng, Yang Chen, Robert Amor, and Michael Witbrock. Robust node classification on graph data with graph and label noise. In *Proceedings of the AAAI conference on artificial intelligence*, 2024.

Shuman Zhuang, Zhihao Wu, Zhaoliang Chen, Hong-Ning Dai, and Ximeng Liu. Refine then classify: Robust graph neural networks with reliable neighborhood contrastive refinement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

A APPENDIX

A.1 ALGORITHMS

Algorithm 1 Overall framework of DEFENCE

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with features $\mathbf{X} \in \mathbb{R}^{N \times d}$, training set $\mathcal{D}_{\text{trn}}$ with partially corrupted labels, validation set $\mathcal{D}_{\text{val}}$, embedding size h; number C of classes, and refresh period T
Output: Parameters $\theta = (\theta_{\text{emb}}, \theta_{\text{cls}})$ of the node classifier

Initialize $\theta_{\text{emb}} = (\mathbf{W}_1 \in \mathbb{R}^{h \times d},\ b_1 \in \mathbb{R}^h)$
Initialize $\theta_{\text{cls}} = (\mathbf{W}_2 \in \mathbb{R}^{C \times h},\ b_2 \in \mathbb{R}^C)$
$\theta \leftarrow (\theta_{\text{emb}}, \theta_{\text{cls}})$
repeat
**** Feature-derived anchors (Sec. 3.1) ****
  Compute decoupled embeddings $\mathbf{Z} \in \mathbb{R}^{N \times h}$ with θ_{emb} ▷ Equation (1)
  if current epoch $t \bmod T = 0$ **then**
    Run k-means on $\mathbf{Z}$ to obtain C centroids $\{\mu_c\}_{c=1}^{C}$
    Compute anchor distribution $\mathbf{R} \in \mathbb{R}^{N \times C}$ ▷ Equation (2)
    Match clusters to classes on $\mathcal{D}_{\text{trn}}$ via Hungarian matching
    Obtain anchor assignments $\hat{k}$ according to $\mathbf{R}$
  end if
**** Anchor-guided representation shaping (Sec. 3.2) ****
  Compute anchor-filtered contrastive loss $\mathcal{L}_{\text{cont}}$ ▷ Equation (3)
  Compute posterior-anchor alignment loss $\mathcal{L}_{\text{align}}$ ▷ Equation (4)
**** Gradient-matched supervision (Sec. 3.3) ****
  for $i \in \mathcal{D}_{\text{trn}}$ **do**
    Compute gradient agreement score α_i w.r.t. θ_{cls}
    Rescale α_i to obtain label reliability ω_i. ▷ Equation (5)
  end for
  Compute reliability-weighted supervised loss $\mathcal{L}_{\text{sup}}$ ▷ Equation (6)
  Update θ minimizing $\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda_{\text{cont}}\mathcal{L}_{\text{cont}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}}$ ▷ Equation (7)
until max epochs reached or early stopping is triggered

A.2 TABLES

Table 3: Symbols.

Symbol	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	input graph with sets of nodes and edges, respectively
$\mathbf{X}$	node feature matrix ($\in \mathbb{R}^{N \times d}$)
N, d	# of nodes, input feature dimension
C	number of classes
$\rho_l,\ \rho_s$	label- / structure-noise rates
$\mathcal{D}_{\text{trn}},\ \mathcal{D}_{\text{val}}$	train / validation node sets
$\theta_{\text{emb}}, \theta_{\text{cls}}$	encoder / classifier parameters with $\theta = \{\theta_{\text{emb}}, \theta_{\text{cls}}\}$
$z_i \in \mathbb{R}^h$	pre-propagation embedding of node i
$\mathbf{Z} \in \mathbb{R}^{N \times h}$	stacked matrix of z_i for all nodes $i \in \mathcal{V}$
$\mu_c \in \mathbb{R}^h$	c-th anchor centroid (computed by K-means)
$r_i \in \mathbb{R}^C$	soft anchor distribution of node i
$\mathbf{R} \in \mathbb{R}^{N \times C}$	matrix of anchor distributions
$\hat{k}_i$	hard anchor id for node i
ω_i	reliability weight of node i derived from α_i
$\mathcal{L}_{\text{cont}}$	anchor-filtered contrastive loss
$\mathcal{L}_{\text{align}}$	posterior–anchor alignment loss
$\mathcal{L}_{\text{sup}}$	reliability-weighted node classification loss
$\lambda_{\text{cont}}, \lambda_{\text{align}}$	balancing parameters for $\mathcal{L}_{\text{cont}}$ and $\mathcal{L}_{\text{align}}$, respectively

Table 4: Summary of datasets.

Dataset	#classes	#nodes	#edges	#features
Cora[1]	7	2,708	10,556	1,433
PubMed[1]	3	19,717	88,648	500
Citeseer-full[2]	6	4,230	10,674	602
Flickr[3]	9	7,575	479,476	12,047

[1] https://github.com/kimiyoung/planetoid
[2] https://github.com/abojchevski/graph2gauss
[3] https://github.com/AnryYang/PANE

A.3 PROOFS

A.3.1 PROOF OF THEOREM 1

Proof. We first quantify how well anchor agreement distinguishes same-class pairs from different-class pairs. Let

$$a = P(A_{uv} = 1 \mid T_{uv} = 1), \qquad b = P(A_{uv} = 1 \mid T_{uv} = 0). \tag{12}$$

When $T_{uv} = 1$, the two nodes share the same clean class. Their anchor assignments agree either when both are assigned to the correct anchor or when both are incorrectly assigned to the same one of the remaining $C - 1$ anchors. Under the anchor error model in Equation (8),

$$a = (1 - \epsilon_{\text{anc}})^2 + \frac{\epsilon_{\text{anc}}^2}{C - 1}. \tag{13}$$

When $T_{uv} = 0$, the two nodes belong to different clean classes. Their anchor assignments agree either when one node is assigned correctly and the other is incorrectly assigned to the same anchor, or when both are incorrectly assigned to the same one of the remaining $C - 2$ anchors. Thus,

$$b = \frac{2\epsilon_{\text{anc}}(1 - \epsilon_{\text{anc}})}{C - 1} + \frac{(C - 2)\epsilon_{\text{anc}}^2}{(C - 1)^2}. \tag{14}$$

Their difference is

$$a - b = \left(1 - \frac{C\epsilon_{\text{anc}}}{C - 1}\right)^2 = \Delta_{\text{anc}}. \tag{15}$$

Since $\epsilon_{\text{anc}} < (C - 1)/C$, we have $\Delta_{\text{anc}} > 0$. Hence, anchor agreement is more likely for same-class pairs than for different-class pairs.

We now consider the positive candidate pool. Before anchor filtering, a pair sampled uniformly from the observed edge set $\mathcal{E}$ belongs to the same clean class with probability h. After retaining only anchor-consistent pairs, Bayes' rule gives

$$p_+ = P(T_{ij} = 1 \mid A_{ij} = 1) = \frac{ha}{ha + (1 - h)b}. \tag{16}$$

Therefore,

$$p_+ - h = \frac{h(1 - h)(a - b)}{ha + (1 - h)b}. \tag{17}$$

Since $ha + (1 - h)b = P(A_{ij} = 1) \leq 1$ and $a - b = \Delta_{\text{anc}}$,

$$p_+ - h \geq h(1 - h)\Delta_{\text{anc}}. \tag{18}$$

Similarly, before anchor filtering, a pair sampled uniformly from $\mathcal{P}$ belongs to different clean classes with probability τ. After retaining only anchor-inconsistent pairs,

$$p_- = P(T_{uv} = 0 \mid A_{uv} = 0) = \frac{\tau(1 - b)}{\tau(1 - b) + (1 - \tau)(1 - a)}. \tag{19}$$

It follows that

$$p_- - \tau = \frac{\tau(1-\tau)(a-b)}{\tau(1-b)+(1-\tau)(1-a)}. \tag{20}$$

Since $\tau(1-b)+(1-\tau)(1-a) = P(A_{uv}=0) \le 1$,

$$p_- - \tau \ge \tau(1-\tau)\Delta_{\text{anc}}. \tag{21}$$

Thus, anchor filtering increases the clean-class reliability of both candidate pools by at least the amounts stated in Equation (9). □

A.3.2 Proof of Theorem 2

Proof. Within each anchor-refresh window, r_i is treated as constant. Therefore, for any posterior $p_\theta(\cdot \mid i)$, the gradient of its alignment term with respect to o_i is

$$\nabla_{o_i} \mathrm{KL}(r_i \,\|\, p_\theta(\cdot \mid i)) = p_\theta(\cdot \mid i) - r_i. \tag{22}$$

For an incorrect class $c \neq y$, the clean- and noisy-label posteriors differ only in their y- and c-components. Consequently,

$$\begin{aligned} &\left\| p_\theta^{\text{noisy},c}(\cdot \mid i) - r_i \right\|_2^2 - \left\| p_\theta^{\text{clean}}(\cdot \mid i) - r_i \right\|_2^2 \\ &= (\bar{s} - r_i(y))^2 + (s - r_i(c))^2 - (s - r_i(y))^2 - (\bar{s} - r_i(c))^2 = 2(s-\bar{s})\big(r_i(y) - r_i(c)\big). \end{aligned} \tag{23}$$

Since $\sum_{c=1}^{C} r_i(c) = 1$, averaging the anchor difference over all incorrect classes gives

$$\frac{1}{C-1}\sum_{c \neq y}\big(r_i(y) - r_i(c)\big) = r_i(y) - \frac{1 - r_i(y)}{C-1} = \frac{C r_i(y) - 1}{C-1}. \tag{24}$$

Combining Equation (22)–Equation (24) and taking the conditional expectation yields

$$\begin{aligned} &\frac{1}{C-1}\sum_{c \neq y} \mathbb{E}\left[\left\| \nabla_{o_i} \mathrm{KL}\Big(r_i \,\|\, p_\theta^{\text{noisy},c}(\cdot \mid i)\Big) \right\|_2^2 - \left\| \nabla_{o_i} \mathrm{KL}\big(r_i \,\|\, p_\theta^{\text{clean}}(\cdot \mid i)\big) \right\|_2^2 \mid y_i^\star = y \right] \\ &\quad = 2(s-\bar{s}) \frac{C\mathbb{E}[r_i(y) \mid y_i^\star = y] - 1}{C-1}. \end{aligned} \tag{25}$$

Because $s > 1/C$, we have $s > \bar{s}$. Moreover, the better-than-random assumption gives $C\mathbb{E}[r_i(y) \mid y_i^\star = y] - 1 > 0$. Hence, the right-hand side of Equation (25) is strictly positive, which proves Equation (10). □

A.3.3 Proof of Theorem 3

Proof. DeFence mainly consists of three parts: (i) computing the pre-propagation embeddings $\mathbf{Z}$ and the anchor distribution $\mathbf{R}$ (lines 4-10 in Algorithm 1), (ii) computing the anchor-filtered contrastive loss $\mathcal{L}_{\text{cont}}$ and the posterior-anchor alignment loss $\mathcal{L}_{\text{align}}$ (lines 11-12 in Algorithm 1), and (iii) computing the reliability-weighted supervised loss $\mathcal{L}_{\text{sup}}$ (lines 13-18 in Algorithm 1).

First, computing $\mathbf{Z}$ with a linear layer costs $\mathcal{O}(dh|\mathcal{V}|)$, where d is the input feature dimension and h is the hidden embedding dimension. Computing anchors via K-Means on $\mathbf{Z}$ with C centroids for I_k iterations, plus soft-membership evaluation, costs $\mathcal{O}\left(I_k C h|\mathcal{V}|\right)$, where C is the number of classes. To align clusters with classes, we build the cluster-class confusion matrix $\mathbf{M}$ on the training split and compute the one-to-one mapping with the Hungarian algorithm. Constructing $\mathbf{M}$ costs $\mathcal{O}(|\mathcal{D}_{\text{trn}}|)$, solving the $C \times C$ linear assignment using the Hungarian algorithm takes $\mathcal{O}(C^3)$, and applying the resulting permutation Π to all anchor distributions costs $\mathcal{O}(C|\mathcal{V}|)$. Thus, the first part of DeFence takes $\mathcal{O}\left(h|\mathcal{V}|(d + I_k C) + C^3\right)$.

Second, computing $\mathcal{L}_{\text{cont}}$ takes $\mathcal{O}(Bh)$, where B is the number of positive and negative node pairs sampled, since we compute dot-product similarities between the selected pairs. To compute the posterior-anchor alignment loss $\mathcal{L}_{\text{align}}$, we apply a class-matching linear layer to $\mathbf{Z}$ and perform

APPNP-based propagation (see Sections 3.1, 3.2). The linear layer costs $\mathcal{O}(Ch|\mathcal{V}|)$, and the APPNP-based propagation costs $\mathcal{O}\left(I_j(|\mathcal{V}| + |\mathcal{E}|)\right)$ where I_j is the number of iterations for computing the PageRank scores. Thus, the second part of DEFENCE takes $\mathcal{O}\left(I_j(|\mathcal{V}| + |\mathcal{E}|) + Bh\right)$ where $\mathcal{O}(Bh)$ is dominated by the first part $\mathcal{O}\left(h|\mathcal{V}|(d + I_k C)\right)$ of DEFENCE.

Lastly, to compute $\mathcal{L}_{\text{sup}}$, we first construct $\mathcal{D}_{\text{ref}}$ by selecting the top 10% most confident non-training nodes. Using the confidence scores obtained from the post-propagation predictions, this selection can be performed with a linear-time order-statistic selection algorithm, costing $\mathcal{O}(|\mathcal{V} \setminus \mathcal{D}_{\text{trn}}|)$. We then compute the reference gradient on $\mathcal{D}_{\text{ref}}$ and per-example gradients on $\mathcal{D}_{\text{trn}}$ (refer to Section 3.3) by a backward pass restricted to the class-matching linear layer. Thus, the third part takes $\mathcal{O}\big(|\mathcal{V} \setminus \mathcal{D}_{\text{trn}}| + Ch(|\mathcal{D}_{\text{ref}}| + |\mathcal{D}_{\text{trn}}|)\big) = \mathcal{O}(Ch|\mathcal{V}|)$. Since it is dominated by the first part of DEFENCE, and since C is fixed with respect to graph size, the additive Hungarian cost $\mathcal{O}(C^3)$ does not change the leading order, the total time complexity of DEFENCE is $\mathcal{O}\left((dh + I_k Ch + I_j)|\mathcal{V}| + I_j|\mathcal{E}|\right)$, which ends the proof. □

A.4 IMPLEMENTATION DETAILS

We provide detailed implementation settings for DEFENCE and the baselines. Unless otherwise specified, we use the authors' released implementations as-is. We keep the public Planetoid splits and the standard preprocessing (row-normalized features). For each setting, we run 5 random seeds and select hyperparameters on the validation set.

GCN (Kipf & Welling, 2017). We use the PyG implementation (Fey & Lenssen, 2019) with a 2-layer backbone (hidden dimension 64). Training uses Adam (Kingma & Ba, 2015) with learning rate 0.01. We apply early stopping with patience 50.

NRGNN (Dai et al., 2021). We use the authors' official implementation without modification. We tune the two loss weights on the author-prescribed grid: $\alpha, \beta \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$.

TFR (Wang et al., 2025). No official implementation is publicly available; we reimplement TFR following the method description and loss formulation in the paper. We adopt the backbone width, dropout, learning rate, and weight decay from the authors' released configuration for their most architecturally similar (citation-network) dataset, and tune the two hyperparameters most tied to noise robustness, the reconstruction loss weight $\beta \in \{0.05, 0.2, 0.5\}$ and the pseudo-label confidence ratio $\in \{0.1, 0.3, 0.5\}$, on a representative validation setting per dataset.

ProCon (Li et al., 2025). No official implementation is publicly available, and the appendix reporting the paper's exact hyperparameter values is not included in the published version; we reimplement ProCon from the equations and method description in the main text. We fix the loss balance weight $\lambda = 0.01$ as reported in the paper, and tune the peer-set size $k \in \{5, 10, 20\}$ and the structure/feature similarity blend $\alpha \in \{0.3, 0.5, 0.7\}$ on a representative validation setting; the remaining hyperparameters (prototype momentum, augmentation ratios) follow standard defaults from the self-supervised learning literature, as the paper does not report them.

GCN-SVD (Entezari et al., 2020). We adopt the GCN-SVD implementation from Deeprobust (Li et al., 2020). We tune the truncated SVD rank k of the adjacency matrix, sweeping $k \in \{5, 10, 15, 50\}$. Other training options follow the default implementation settings.

GNNGuard (Zhang & Zitnik, 2020). We use the GNNGuard implementation from Deeprobust (Li et al., 2020) while adopting GCN (Kipf & Welling, 2017) due to its robust and strong performance.

RGCN (Zhu et al., 2019). We use the RGCN implementation from Deeprobust (Li et al., 2020) with default settings.

NoisyGNN (Dai et al., 2022). We use the official NoisyGNN implementation from the authors. We perform a one-dimensional grid search over the noise ratio parameter $\beta \in [0, 1.0)$ with step 0.01 as described in the original paper.

RNCGLN (Zhu et al., 2024). We use the authors' official implementation without modification and keep all method-specific hyperparameters at their released defaults.

GDF (Zhang et al., 2026). No official implementation is publicly available; we reimplement GDF following the equations in the paper. Following the paper, we fix $d_\mu = 0.7$ and tune the two

regularizer weights $\lambda_{cc}, \lambda_{sc} \in \{0, 0.2, 0.4, 0.5, 0.6, 0.8, 1.0\}$, the same grid used in the paper's own ablation tables.

DEFENCE (proposed). We implement our method on top of PyTorch Geometric (Fey & Lenssen, 2019). We tune only the two method-specific hyperparameters: λ_{cont} and λ_{align} in Equation (7). Specifically, we sweep $\lambda_{\text{cont}} \in \{0.05, 0.10, 0.20, 0.30, 0.50\}$ and $\lambda_{\text{align}} \in \{0.10, 0.20, 0.30, 0.50, 1.00\}$. The training schedule matches that of GCN: Adam (learning rate 0.01) with early stopping (patience 50, 100-epoch warm-up). For APPNP, we set $K = 10$ propagation steps and teleport $\alpha = 0.1$. During training we periodically form soft clusters from the hidden embeddings every 10 epochs. The number B of positive and negative node pairs in Equation (3) is set to 1024 and the margin m for negative samples is set to 0.5.

A.5 RESULTS UNDER EXTREME NOISE RATIOS

Table 5: Classification accuracy under extreme label-structure noise ratios. Best results are in bold. O.O.M. denotes out-of-memory error.

Datasets	$\{\rho_l, \rho_s\}$	GCN	NRGNN	TFR	ProCon	GCN-SVD	GNNGuard	RGCN	NoisyGNN	RNCGLN	GDF	DEFENCE (ours)
Cora	$\{0.5, 0.1\}$	55.2 ± 4.0	41.3 ± 9.7	56.5 ± 2.7	50.7 ± 1.5	36.7 ± 0.5	40.5 ± 5.0	53.0 ± 2.9	52.9 ± 3.8	52.2 ± 0.0	41.2 ± 2.4	**60.8 ± 4.4**
	$\{0.1, 0.5\}$	64.9 ± 2.3	42.6 ± 2.8	53.2 ± 1.4	44.4 ± 4.1	54.2 ± 0.9	62.6 ± 2.7	64.2 ± 3.8	60.8 ± 1.7	69.3 ± 0.0	58.3 ± 4.1	**71.0 ± 1.9**
PubMed	$\{0.5, 0.1\}$	51.3 ± 8.8	49.0 ± 5.2	54.1 ± 3.1	50.4 ± 7.5	43.4 ± 5.2	54.9 ± 3.7	52.2 ± 7.4	**57.9 ± 9.3**	O.O.M.	45.8 ± 7.0	53.9 ± 5.0
	$\{0.1, 0.5\}$	65.5 ± 2.7	60.0 ± 4.6	63.2 ± 5.1	58.2 ± 1.0	66.5 ± 0.9	70.0 ± 1.3	65.8 ± 3.1	68.5 ± 2.7	O.O.M.	63.8 ± 1.7	**74.5 ± 1.6**
Citeseer-full	$\{0.5, 0.1\}$	69.1 ± 1.2	65.8 ± 2.1	**75.4 ± 1.3**	66.6 ± 1.5	54.9 ± 0.5	58.1 ± 0.7	62.4 ± 1.7	68.3 ± 1.5	51.8 ± 0.0	44.9 ± 1.1	72.7 ± 2.9
	$\{0.1, 0.5\}$	72.7 ± 0.3	71.2 ± 2.0	71.0 ± 1.4	60.4 ± 1.1	69.4 ± 1.9	71.9 ± 0.7	73.4 ± 0.8	73.8 ± 1.0	76.1 ± 0.0	65.4 ± 1.4	**76.7 ± 1.4**
Flickr	$\{0.5, 0.1\}$	44.5 ± 1.7	N/A	21.9 ± 1.3	32.4 ± 2.0	**49.4 ± 2.1**	41.0 ± 1.5	24.1 ± 4.0	42.0 ± 0.4	O.O.M.	36.1 ± 2.0	49.1 ± 1.2
	$\{0.1, 0.5\}$	43.1 ± 6.2	N/A	17.4 ± 1.1	33.1 ± 0.9	62.9 ± 1.1	65.5 ± 3.2	13.0 ± 1.9	52.4 ± 1.1	O.O.M.	53.2 ± 2.0	**72.4 ± 0.7**

We evaluate DEFENCE under more severe joint corruption: we additionally test extreme noise settings where either label noise or structure noise increases to 0.5. Table 5 shows that DEFENCE achieves the best performance in extreme noise settings. This shows that the feature-derived anchors provide stable guidance even when labels or edges are highly corrupted.

A.6 RESULTS UNDER PAIRWISE LABEL NOISE

Table 6: Classification accuracy under pairwise label noise and structure noise. Best results are in bold and second-best results are underlined. O.O.M. denotes out-of-memory error.

Datasets	$\{\rho_l, \rho_s\}$	GCN	NRGNN	TFR	ProCon	GCN-SVD	GNNGuard	RGCN	NoisyGNN	RNCGLN	GDF	DEFENCE (ours)
Cora	$\{0.1, 0.1\}$	72.5 ± 2.2	63.8 ± 8.6	67.7 ± 4.5	53.2 ± 2.4	55.9 ± 1.4	63.8 ± 1.8	72.4 ± 0.8	71.6 ± 1.7	71.0 ± 0.0	68.5 ± 1.0	**78.0 ± 2.2**
	$\{0.1, 0.2\}$	69.5 ± 0.7	59.4 ± 7.4	62.6 ± 4.9	49.5 ± 4.7	56.9 ± 0.3	65.4 ± 1.1	70.4 ± 1.2	68.9 ± 0.7	69.4 ± 0.0	63.9 ± 0.8	**75.9 ± 0.5**
	$\{0.2, 0.1\}$	65.4 ± 0.6	55.2 ± 4.3	57.5 ± 4.7	50.9 ± 1.7	53.3 ± 0.9	57.5 ± 1.1	67.1 ± 2.3	66.1 ± 1.9	68.5 ± 0.0	62.3 ± 0.7	**72.2 ± 1.7**
	$\{0.2, 0.2\}$	63.1 ± 1.0	51.0 ± 4.0	51.9 ± 3.1	48.3 ± 1.7	54.4 ± 1.3	57.6 ± 1.3	63.5 ± 1.8	64.1 ± 1.5	69.4 ± 0.0	58.2 ± 2.4	**71.9 ± 1.8**
PubMed	$\{0.1, 0.1\}$	71.8 ± 2.4	71.7 ± 1.8	69.2 ± 2.0	64.0 ± 2.2	66.4 ± 0.6	72.1 ± 2.6	72.9 ± 1.8	70.4 ± 1.7	O.O.M.	69.6 ± 2.8	**74.8 ± 1.3**
	$\{0.1, 0.2\}$	70.2 ± 1.5	68.2 ± 3.5	69.1 ± 3.4	62.8 ± 2.1	66.7 ± 0.3	71.1 ± 1.6	70.2 ± 1.0	69.7 ± 0.4	O.O.M.	68.3 ± 1.0	**73.4 ± 2.5**
	$\{0.2, 0.1\}$	66.5 ± 3.3	66.9 ± 2.0	66.7 ± 2.0	63.1 ± 0.6	61.4 ± 0.7	66.7 ± 2.7	66.9 ± 1.4	68.8 ± 2.4	O.O.M.	62.8 ± 4.0	**71.4 ± 2.4**
	$\{0.2, 0.2\}$	65.1 ± 2.3	59.6 ± 4.0	62.4 ± 1.9	60.7 ± 1.2	62.3 ± 1.1	67.1 ± 2.1	65.4 ± 1.6	67.5 ± 1.8	O.O.M.	61.6 ± 3.5	**69.9 ± 1.6**
Citeseer-full	$\{0.1, 0.1\}$	81.6 ± 0.8	81.0 ± 1.1	**82.5 ± 0.5**	69.2 ± 1.9	70.4 ± 1.7	73.2 ± 0.7	82.0 ± 0.4	81.1 ± 0.5	77.6 ± 0.0	74.7 ± 0.3	81.8 ± 2.0
	$\{0.1, 0.2\}$	78.5 ± 0.7	78.1 ± 0.8	79.5 ± 1.1	65.9 ± 0.8	70.0 ± 0.8	73.1 ± 1.5	79.0 ± 0.7	78.3 ± 0.9	77.4 ± 0.0	71.3 ± 0.3	**80.0 ± 1.4**
	$\{0.2, 0.1\}$	77.8 ± 1.9	78.0 ± 1.4	**80.7 ± 1.1**	68.4 ± 2.6	67.6 ± 1.6	71.2 ± 0.9	77.3 ± 0.3	77.3 ± 1.3	72.8 ± 0.0	67.3 ± 0.7	80.0 ± 1.6
	$\{0.2, 0.2\}$	75.2 ± 0.7	75.2 ± 1.2	**78.1 ± 1.2**	62.3 ± 5.2	67.0 ± 1.7	70.9 ± 0.8	75.0 ± 1.6	74.5 ± 0.9	74.4 ± 0.0	64.3 ± 0.6	76.9 ± 2.7
Flickr	$\{0.1, 0.1\}$	54.5 ± 2.1	43.7 ± 2.5	25.0 ± 1.3	36.5 ± 2.7	64.2 ± 0.6	65.9 ± 2.6	30.6 ± 0.7	54.3 ± 0.8	O.O.M.	54.0 ± 0.6	**75.2 ± 1.8**
	$\{0.1, 0.2\}$	49.6 ± 5.5	43.4 ± 2.0	20.4 ± 0.3	36.2 ± 1.5	63.6 ± 0.6	65.6 ± 3.0	22.4 ± 2.1	54.2 ± 0.4	O.O.M.	54.3 ± 0.6	**74.8 ± 1.0**
	$\{0.2, 0.1\}$	51.6 ± 0.6	43.8 ± 2.2	26.6 ± 2.3	34.5 ± 2.3	59.7 ± 1.0	59.7 ± 2.4	29.8 ± 2.6	51.0 ± 0.4	O.O.M.	49.1 ± 0.2	**69.0 ± 1.6**
	$\{0.2, 0.2\}$	48.7 ± 0.8	43.4 ± 1.7	20.7 ± 3.2	36.3 ± 1.2	59.5 ± 0.4	60.6 ± 2.9	20.5 ± 2.4	50.2 ± 0.5	O.O.M.	49.9 ± 0.3	**68.2 ± 0.7**

We evaluate DEFENCE under pairwise label noise. Table 6 shows that DEFENCE achieves the best performance in pairwise label-noise settings. These results indicate that DEFENCE is robust not only to random label flips but also to systematic class-level corruption.

A.7 RESULTS UNDER MISSING-EDGE STRUCTURE NOISE

In the main experiments, we inject spurious edges to generate structure noise, which is a common setting for sparse graphs (random edge flips are often addition-dominant in practice). To further verify robustness under structural corruption, we also evaluate missing-edge noise by randomly removing a fraction ρ_s of observed edges.

Table 7 reports the results. Compared with the addition-based noise in Table 1, missing-edge noise generally causes smaller performance drops, suggesting that random edge removal is less harmful

Table 7: Classification accuracy under missing-edge structure noise. Best results are in bold and second-best results are underlined. O.O.M. denotes out-of-memory error.

Datasets	$\{\rho_l, \rho_s\}$	GCN	NRGNN	TFR	ProCon	GCN-SVD	GNNGuard	RGCN	NoisyGNN	RNCGLN	GDF	DEFENCE (ours)
Cora	$\{0.1, 0.1\}$	77.1 ± 0.9	70.5 ± 2.2	75.1 ± 1.4	54.6 ± 5.5	58.1 ± 0.6	65.2 ± 1.2	77.6 ± 0.1	76.8 ± 0.6	76.2 ± 0.0	71.5 ± 1.4	**79.6 ± 1.4**
	$\{0.1, 0.2\}$	75.9 ± 1.0	69.5 ± 1.8	74.5 ± 0.5	55.5 ± 5.0	54.8 ± 2.6	62.2 ± 2.6	75.8 ± 1.1	75.1 ± 0.3	76.4 ± 0.0	71.3 ± 0.3	**79.1 ± 1.4**
	$\{0.2, 0.1\}$	71.8 ± 2.4	63.9 ± 4.2	70.6 ± 1.5	57.9 ± 2.6	54.4 ± 0.9	58.7 ± 3.9	71.8 ± 0.7	72.6 ± 0.8	76.8 ± 0.0	64.6 ± 1.4	**77.7 ± 2.1**
	$\{0.2, 0.2\}$	70.8 ± 2.5	62.7 ± 4.2	71.5 ± 2.2	54.7 ± 5.3	50.9 ± 0.4	58.4 ± 2.3	70.9 ± 0.6	70.4 ± 0.4	75.1 ± 0.0	64.3 ± 1.4	**77.4 ± 1.2**
PubMed	$\{0.1, 0.1\}$	74.8 ± 1.6	73.7 ± 2.1	75.0 ± 1.1	65.4 ± 1.1	66.8 ± 0.6	73.2 ± 1.3	75.4 ± 2.2	74.1 ± 3.1	O.O.M.	72.7 ± 2.0	**78.4 ± 2.0**
	$\{0.1, 0.2\}$	74.1 ± 2.5	72.5 ± 1.7	74.3 ± 2.5	63.7 ± 2.5	66.5 ± 1.7	72.3 ± 2.0	74.5 ± 2.3	73.3 ± 2.9	O.O.M.	71.3 ± 3.5	**76.6 ± 2.0**
	$\{0.2, 0.1\}$	70.9 ± 2.5	64.3 ± 4.5	71.9 ± 1.1	63.7 ± 1.9	65.0 ± 0.9	68.9 ± 1.6	70.1 ± 1.2	72.4 ± 1.2	O.O.M.	67.0 ± 1.0	**74.8 ± 1.6**
	$\{0.2, 0.2\}$	70.7 ± 3.3	63.6 ± 5.1	71.7 ± 1.7	63.3 ± 1.0	60.8 ± 4.2	68.7 ± 2.7	68.0 ± 1.2	70.3 ± 2.0	O.O.M.	66.2 ± 0.8	**74.1 ± 1.9**
Citeseer	$\{0.1, 0.1\}$	84.4 ± 0.9	84.1 ± 1.0	85.7 ± 0.6	71.0 ± 0.9	71.1 ± 0.5	73.6 ± 1.4	85.3 ± 0.4	84.5 ± 0.6	80.1 ± 0.0	78.5 ± 0.7	**86.6 ± 1.3**
	$\{0.1, 0.2\}$	83.3 ± 1.2	82.9 ± 1.2	84.9 ± 0.9	69.8 ± 0.5	70.8 ± 0.6	73.2 ± 1.4	83.6 ± 0.8	83.5 ± 0.9	79.3 ± 0.0	77.2 ± 0.6	**85.5 ± 1.4**
	$\{0.2, 0.1\}$	82.3 ± 1.0	81.6 ± 0.7	85.3 ± 0.4	71.5 ± 0.7	69.0 ± 0.2	71.1 ± 2.7	82.0 ± 0.9	81.6 ± 1.1	78.1 ± 0.0	71.7 ± 1.5	**85.4 ± 0.9**
	$\{0.2, 0.2\}$	81.0 ± 1.1	80.4 ± 1.2	**84.4 ± 0.6**	68.7 ± 0.9	67.7 ± 0.9	69.5 ± 2.5	80.1 ± 1.3	81.4 ± 1.1	77.0 ± 0.0	70.1 ± 1.9	84.1 ± 1.3
Flickr	$\{0.1, 0.1\}$	60.2 ± 1.5	43.5 ± 2.0	40.5 ± 1.0	39.0 ± 2.5	64.3 ± 0.1	65.5 ± 3.6	35.9 ± 1.8	58.4 ± 0.2	O.O.M.	58.5 ± 0.5	**77.1 ± 1.6**
	$\{0.1, 0.2\}$	60.5 ± 0.6	43.4 ± 1.7	38.4 ± 2.3	41.0 ± 1.1	64.9 ± 1.1	66.3 ± 3.0	37.1 ± 0.7	58.1 ± 0.4	O.O.M.	57.3 ± 0.8	**76.3 ± 0.9**
	$\{0.2, 0.1\}$	59.0 ± 1.7	43.4 ± 1.8	40.1 ± 1.6	37.6 ± 2.2	61.3 ± 1.2	62.3 ± 4.6	38.5 ± 3.7	57.2 ± 0.7	O.O.M.	54.3 ± 1.0	**72.2 ± 2.1**
	$\{0.2, 0.2\}$	58.2 ± 0.6	43.3 ± 1.5	40.3 ± 2.2	38.7 ± 2.5	62.0 ± 1.4	60.6 ± 3.2	38.6 ± 1.4	55.1 ± 0.6	O.O.M.	53.3 ± 0.4	**71.7 ± 1.5**

than adding spurious edges. Nevertheless, DEFENCE achieves the best performance in most settings, confirming its robustness to both spurious-edge and missing-edge structure noise.

A.8 RESULTS WITH DIFFERENT BACKBONES

In the main experiments, we instantiate DEFENCE with APPNP, which decouples feature transformation from graph propagation. To examine whether the performance of DEFENCE depends on this particular backbone, we additionally instantiate DEFENCE with two other decoupled GNN backbones, GPR-GNN and DAGNN, while keeping the remaining components and experimental protocol unchanged.

Table 8 reports the results. All three backbones achieve comparable performance across the joint label-structure noise settings, with average accuracies of 75.27%, 75.00%, and 74.20% for APPNP, GPR-GNN, and DAGNN, respectively. APPNP achieves the highest overall average accuracy, while GPR-GNN performs competitively and yields better results in several settings on Citeseer and Flickr. These results demonstrate that DEFENCE is not tied to the specific propagation mechanism.

Table 8: Node classification accuracy of DEFENCE with different backbone architectures under joint label-structure noise. The last row gives the average across all datasets and noise settings.

Datasets	$\{\rho_l, \rho_s\}$	**APPNP**	**GPR-GNN**	**DAGNN**
Cora	{0.1, 0.1}	79.0 ± 2.2	77.0 ± 1.8	76.5 ± 2.3
	{0.1, 0.2}	75.4 ± 1.2	72.8 ± 1.7	73.7 ± 2.0
	{0.2, 0.1}	73.7 ± 0.1	74.0 ± 2.1	73.0 ± 0.3
	{0.2, 0.2}	72.6 ± 1.4	70.9 ± 2.8	69.8 ± 0.1
PubMed	{0.1, 0.1}	75.0 ± 2.4	73.1 ± 1.7	74.8 ± 0.7
	{0.1, 0.2}	73.9 ± 1.5	72.7 ± 1.6	72.0 ± 0.2
	{0.2, 0.1}	71.7 ± 0.7	69.8 ± 2.9	70.8 ± 1.9
	{0.2, 0.2}	69.9 ± 0.7	67.3 ± 2.9	70.7 ± 1.5
Citeseer	{0.1, 0.1}	83.1 ± 1.8	84.0 ± 0.9	82.8 ± 1.1
	{0.1, 0.2}	79.7 ± 1.0	81.7 ± 1.5	80.6 ± 0.7
	{0.2, 0.1}	81.7 ± 1.1	83.0 ± 0.7	82.1 ± 1.6
	{0.2, 0.2}	78.8 ± 1.8	79.5 ± 1.4	79.3 ± 0.8
Flickr	{0.1, 0.1}	75.1 ± 0.5	76.1 ± 1.5	74.1 ± 1.9
	{0.1, 0.2}	74.3 ± 1.9	75.4 ± 0.4	72.8 ± 3.1
	{0.2, 0.1}	71.5 ± 1.4	71.5 ± 1.5	68.0 ± 2.8
	{0.2, 0.2}	69.0 ± 2.3	71.0 ± 0.7	66.2 ± 3.6
Average		75.27	75.00	74.20