STEALTHY YET EFFECTIVE: IN-DISTRIBUTION NODE INJECTION ATTACKS FOR HYPEREDGE PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep hypergraph models are widely adopted in real-world decision-making systems due to their powerful higher-order modeling capabilities for complex data, making their security and robustness increasingly critical. Recent studies have revealed adversarial vulnerabilities in deep hypergraph models on node-level tasks that predict individual node properties. However, their robustness in hyperedge prediction, a fundamental task for inferring missing or future group-level interactions among multiple entities, remains largely unexplored. To fill this gap, we systematically analyze mainstream injection-based attacks and identify two key challenges in applying them to hyperedge prediction: (1) *Perturbation attenuation*: we theoretically demonstrate that group-level aggregation in edge prediction weakens the adversarial influence of injected nodes, limiting attack effectiveness. (2) *Distributional deviation*: empirical results reveal pronounced shifts in membership- and hyperedge-level statistical distributions after injection, compromising attack stealthiness. To tackle these challenges, we propose DPHIA, the first effective and unnoticeable node injection attack tailored for hyperedge prediction. DPHIA employs an adaptive ranking-based objective to perturb the decision boundary between positive and negative hyperedges, thereby misleading the predictor. A dual-level adversarial regularizer constrains distributional shifts in higher-order relations through membership- and hyperedge-level adversarial training, improving attack stealthiness. Extensive experiments on five real-world datasets demonstrate that DPHIA achieves a superior trade-off between attack effectiveness and detectability compared with state-of-the-art baselines. Our code is available at `https://anonymous.4open.science/r/DPHIA-063D`.

1 INTRODUCTION

Hypergraphs provide a natural framework for modeling higher-order interactions prevalent in complex systems, where each hyperedge can connect more than two entities simultaneously, thereby overcoming the pairwise limitation of conventional graphs (Xia et al., 2021; Behrouz et al., 2023; Kim et al., 2024). Deep hypergraph learning (Li et al., 2026a) has emerged as a foundational paradigm for learning from hypergraph-structured data. By capturing group-level associations through higher-order message passing, it has achieved remarkable performance in diverse applications, such as social recommendation (Yu et al., 2021), fraud detection (Ou et al., 2026), and drug interaction prediction (Saifuddin et al., 2023). As deep hypergraph models are increasingly adopted in real-world decision-making systems, their security and robustness have drawn growing attention.

Recent studies have shown that deep hypergraph models are vulnerable to adversarial attacks (Shi et al., 2025; Arafat et al., 2026; He et al., 2026), posing significant risks in high-stakes domains such as finance, healthcare, and cybersecurity. Existing research has primarily focused on node injection attacks (Chen et al., 2022; Shi et al., 2025; He et al., 2026), a stealthy adversarial approach that introduces new nodes into hypergraphs without altering the original structure. IE-Attack (He et al., 2025) and HyperNear (Cai et al., 2025) represent early attempts at hypergraph node injection, using homophily-preserving malicious node generation to improve both stealth and attack effectiveness. Furthermore, TH-Attack (He et al., 2026) injects semantically divergent nodes into only pivotal hyperedges, significantly enhancing cross-architecture transferability of the injection attack.

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.

However, these attack studies primarily target node classification, leaving the adversarial robustness of hyperedge prediction largely unexplored. Unlike node-level tasks that predict individual properties, hyperedge prediction, a fundamental task in hypergraph learning, aims to infer missing or future relations among groups of entities (Ko et al., 2025; Yu et al., 2025; Li et al., 2026b;c). Robust and trustworthy hyperedge prediction is crucial for real-world business. For example, in insurance risk management, multiple parties may collude to submit fraudulent claims (Chen et al., 2019). Accounts can be modeled as nodes, with hyperedges capturing their joint participation in claims or transactions. A deep hyperedge predictor can infer unobserved group-wise relations to help identify fraudulent groups and guide intervention. By injecting malicious accounts into observed interactions before training, an attacker could degrade predictions across unseen groups, allowing fraudulent claims to evade detection. Such failures could lead to substantial financial losses. These risks highlight the need to understand the vulnerability of hyperedge prediction to adversarial attacks, thereby facilitating the development of more resilient deep hypergraph models.

To fill this gap, we present the first systematic study of node injection attacks for hyperedge prediction. Figure 1 gives an illustration of this setting, where an attacker injects malicious nodes into observed hyperedges before model training. The attack aims to mislead predictions on unseen node groups, causing genuine hyperedges to be missed and negative candidates to be falsely identified as hyperedges. Through comprehensive analyses, we demonstrate that existing node injection attacks face two key challenges in hyperedge prediction: (1) *Perturbation attenuation*. Our theoretical analysis (Section 3.1) shows that group-level aggregation can attenuate the adversarial effects of node perturbations, which makes it harder to disrupt hyperedge predictions. (2) *Distributional deviation*. To investigate whether node injection attacks induce distributional deviations in higher-order relations, we introduce two simple deviation metrics at the membership and hyperedge levels. Our empirical results (Section 3.2) show that higher-order relations involving injected nodes exhibit pronounced distributional deviations from benign ones, making them easier to detect and compromising attack stealthiness.

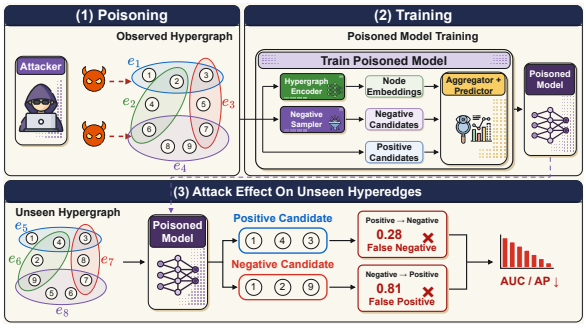


Figure 1: Node injection for hyperedge prediction.

To address these challenges, we propose DPHIA (**D**istribution-**P**reserving **H**ypergraph **I**njection **A**ttack), the first node injection framework for hyperedge prediction. We begin with *Uncertainty-Guided Poisoning Initialization*, which selects a small subset of hyperedges near the decision boundary as injection targets, facilitating subtle yet effective node injection. To refine the malicious node generator, we introduce *Distribution-Preserving Feature Optimization*, which combines: (i) an adaptive ranking-based attack objective, which promotes ranking reversals between positive and negative hyperedges and prioritizes candidate pairs whose ordering is easier to reverse to weaken edge-level separability; and (ii) dual-level adversarial regularization, which uses membership- and hyperedge-level discriminators to constrain distributional deviations in higher-order connectivity. Through adversarial training with a surrogate classifier, DPHIA enables effective and stealthy node injection. Our main contributions can be summarized as follows:

- We present the first systematic study of node injection attacks for hyperedge prediction. Our analysis reveals two key limitations of existing attacks: perturbation attenuation during candidate aggregation and distributional deviations in higher-order relations, which affect attack effectiveness and stealthiness, respectively.
- We introduce DPHIA, a novel node injection attack framework for hyperedge prediction. It combines uncertainty-guided injection target selection with distribution-preserving feature optimization based on an adaptive ranking-based attack objective and dual-level adversarial regularization to achieve effective and stealthy attacks.
- Extensive experiments demonstrate that DPHIA achieves state-of-the-art attack performance while substantially reducing detectability at both the membership and hyperedge levels, achieving a strong trade-off between stealth and efficacy.

2 PRELIMINARIES

2.1 HYPEREDGE PREDICTION TASK

Notations. Let $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ denote a hypergraph, where $\mathcal{V} = \{v_i\}_{i=1}^{|\mathcal{V}|}$ is the node set, $\mathcal{E} = \{e_j\}_{j=1}^{|\mathcal{E}|}$ is the hyperedge set, and each hyperedge $e_j \in \mathcal{E}$ is a nonempty subset of $\mathcal{V}$. The matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times F}$ contains the F-dimensional node features. The hypergraph topology is represented by an incidence matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{E}|}$, where $\mathbf{H}_{ij} = 1$ if $v_i \in e_j$ and $\mathbf{H}_{ij} = 0$ otherwise.

Hyperedge Prediction. Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, we denote $\mathcal{E}' \subseteq 2^{\mathcal{V}} \setminus \mathcal{E}$ as the target set, which typically consists of unobserved hyperedges or new hyperedges that will arrive in the future. Each nonempty element of $2^{\mathcal{V}} \setminus \mathcal{E}$ is referred to as a hyperedge candidate, denoted by c, as it may belong to $\mathcal{E}'$. The hyperedge prediction task aims to train a hyperedge predictor $f_\theta : c \mapsto \{0,1\}$ to predict whether a candidate c belongs to the target set $\mathcal{E}'$. In practice, the predictor is trained to distinguish positive samples in $\mathcal{E}$ from negative samples in $\mathcal{E}^- = \mathrm{NS}(\mathcal{H})$, which are constructed from unobserved node sets through negative sampling (NS) (Ko et al., 2025).

2.2 UNNOTICEABLE NODE INJECTION ATTACK ON HYPEREDGE PREDICTION TASK

We study node injection attacks, where an adversary injects new nodes into the observed hypergraph $\mathcal{H}$ to compromise model training, which can be formulated as:

Definition 1 (Node Injection Attack on Hypergraph)**.** *Given an observed hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, the adversary constructs a poisoned hypergraph $\widehat{\mathcal{H}} = (\mathcal{V} \cup \mathcal{V}_m, \widehat{\mathcal{E}}, \widehat{\mathbf{X}})$, where $\widehat{\mathcal{E}}$ contains the same hyperedges as $\mathcal{E}$, with their memberships augmented by the injected nodes $\mathcal{V}_m$. Accordingly, the incidence and feature matrices become*

$$\widehat{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \mathbf{H}_m \end{bmatrix}, \qquad \widehat{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \mathbf{X}_m \end{bmatrix}. \tag{1}$$

$\mathbf{X}_m \in \mathbb{R}^{|\mathcal{V}_m| \times F}$ denotes the injected node features, while $\mathbf{H}_m \in \{0,1\}^{|\mathcal{V}_m| \times |\mathcal{E}|}$ encodes their memberships to the observed hyperedges. All original node features and incidences remain unchanged.

Attacker's Goal. The objective of the attacker in hyperedge prediction is twofold: (1) The model trained on the poisoned hypergraph should make incorrect predictions on unseen target hyperedges $\mathcal{E}'$; (2) The poisoned hypergraph should not exhibit significant distributional deviations from benign data, thereby ensuring high stealthiness.

Attacker's Knowledge and Capability. We consider a realistic black-box setting: (1) **Black-box knowledge**: the attacker has access only to the observed hypergraph $\mathcal{H}$, but has no knowledge of the victim architecture, parameters, or negative sampling strategy; (2) **Limited poisoning capability**: the attacker may inject at most B malicious nodes and determine their features and connections to observed hyperedges, without modifying any original node features or existing incidences.

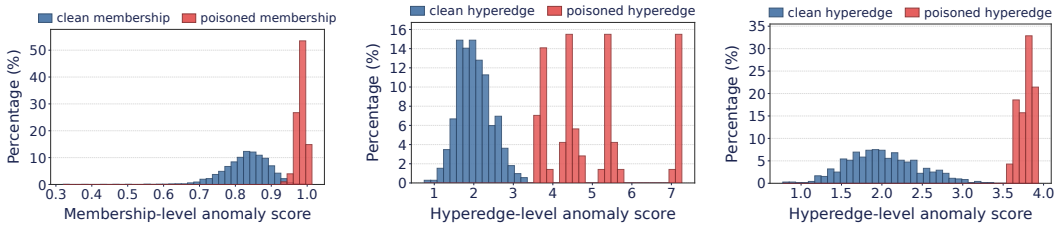
3 ANALYSIS OF ADVERSARIAL ATTACKS ON HYPEREDGE PREDICTION

In this section, we investigate current node injection attacks for hyperedge prediction and reveal two key challenges concerning attack effectiveness and stealthiness.

3.1 EFFECTIVENESS CHALLENGE: PERTURBATION ATTENUATION

Prior studies show that small perturbations can disrupt hypergraph propagation, facilitating attacks on node classification (Cai et al., 2025; Arafat et al., 2026). However, in hyperedge prediction, these perturbed embeddings are further aggregated into candidate representations before scoring. This aggregation can attenuate the adversarial influence of node perturbations, making it harder to compromise hyperedge predictions. To analyze this effect, we first bound the resulting shifts in candidate representations and then examine the perturbations required to degrade hyperedge predictions.

Let $\mathbf{z}_v$ and $\widehat{\mathbf{z}}_v$ denote the embeddings of an original node v obtained through hypergraph message passing before and after injection, respectively. For a hyperedge candidate $c \subseteq \mathcal{V}$, let



(a) TH-Attack (MD). (b) FGA (HD). (c) HyperNear (HD).

Figure 2: Anomaly-score distributions under representative attacks on Cora.

$\mathbf{q}_c = \mathrm{agg}(\{\mathbf{z}_v : v \in c\})$ denote its representation and $\widehat{\mathbf{q}}_c$ its counterpart after injection. To simplify the analysis, we consider max–min pooling, defined by $[\mathbf{q}_c]_j = \max_{v\in c} z_{v,j} - \min_{v\in c} z_{v,j}$. Let $\boldsymbol{\delta}_v = \widehat{\mathbf{z}}_v - \mathbf{z}_v$ denote the embedding perturbation of node v induced by injection. For candidate c, the mean perturbation is $\mathbf{a}_c = |c|^{-1}\sum_{v\in c}\boldsymbol{\delta}_v$. We use $\varepsilon_c = \max_{v\in c}\|\boldsymbol{\delta}_v - \mathbf{a}_c\|_\infty$ to measure the maximum difference between individual node perturbations and their mean within c.

Lemma 1 (Perturbation attenuation under candidate aggregation)**.** *For any candidate c, the shift in its aggregated representation is bounded by*

$$\|\widehat{\mathbf{q}}_c - \mathbf{q}_c\|_\infty \le 2\varepsilon_c. \tag{2}$$

Lemma 1 shows that changes in candidate representations are bounded by differences among node perturbations and vanish when all nodes are perturbed identically. Thus, even large node perturbations can have limited effects on candidate representations when members undergo similar changes. We next study how these representation shifts affect the prediction scores of hyperedge candidates.

Corollary 1 (Ranking stability under perturbation attenuation)**.** *Consider a positive–negative candidate pair (c^+, c^-) and a fixed prediction head g that maps candidate representations to hyperedge probabilities. Assume that g is L_a-Lipschitz with respect to $\|\cdot\|_\infty$, where $L_a > 0$. Let $p_c = g(\mathbf{q}_c)$ and $\widehat{p}_c = g(\widehat{\mathbf{q}}_c)$ denote the predicted hyperedge probabilities of candidate c before and after injection, respectively. For a pair correctly predicted before injection, the initial probability gap is $\Delta = p_{c^+} - p_{c^-} > 0$. To disrupt this ranking, the attacker seeks $\widehat{\Delta} = \widehat{p}_{c^+} - \widehat{p}_{c^-} \le 0$, which requires*

$$\varepsilon_{c^+} + \varepsilon_{c^-} \ge \frac{\Delta}{2L_a}. \tag{3}$$

Thus, a larger initial gap requires greater differences among node perturbations within the two candidates. If this condition is not met, the positive candidate remains ranked above the negative.

Proofs of Lemma 1 and Corollary 1 are provided in Appendix A.

Remark. These results show that candidate aggregation can attenuate node perturbations, making it harder to overcome the initial positive–negative probability gap. Consequently, large node-level perturbations alone do not guarantee effective attacks. This motivates us to directly optimize relative candidate scores after aggregation and focus on positive–negative pairs with smaller probability gaps, in which reversing the ordering requires smaller score perturbations.

3.2 Stealthiness Challenge: Distribution Deviation on Higher-order Relation

In this part, we investigate two types of higher-order distributional deviations that may arise from node injection attacks: membership-level deviation between injected nodes and original members of the same hyperedge, and hyperedge-level deviation between poisoned and benign hyperedges. To quantify these deviations, we introduce two complementary metrics as follows:

Membership-level deviation (MD). For each membership (v, e) with $v \in e$ and $|e| > 1$, we quantify the node's feature deviation from the other members as

$$a_{\mathrm{M}}(v, e) = 1 - \cos(\mathbf{x}_v, \boldsymbol{\mu}_{e\setminus\{v\}}), \qquad \boldsymbol{\mu}_{e\setminus\{v\}} = \frac{1}{|e|-1}\sum_{u\in e\setminus\{v\}} \mathbf{x}_u, \tag{4}$$

where $\cos(\cdot,\cdot)$ denotes cosine similarity and $\boldsymbol{\mu}_{e\setminus\{v\}}$ is the mean feature of the other members. A larger $a_{\mathrm{M}}(v, e)$ indicates a higher likelihood that membership (v, e) is anomalous (Feng et al., 2025).

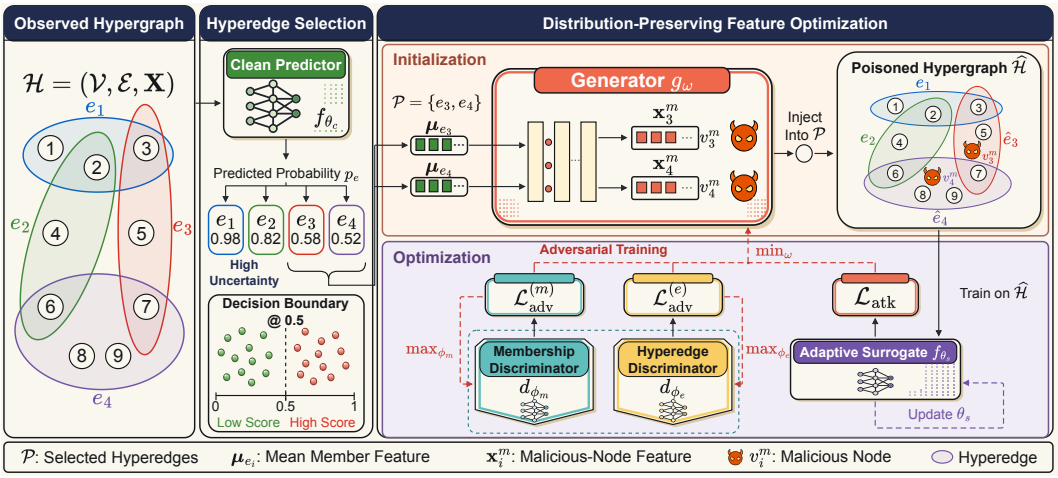


Figure 3: Overview of DPHIA.

Hyperedge-level deviation (HD). Following reconstruction-based outlier detection (Ding et al., 2019), we quantify hyperedge-level attribute deviation by averaging the reconstruction errors of its members. For each hyperedge e, the deviation score is computed as

$$a_{\mathrm{H}}(e) = \frac{1}{|e|} \sum_{v \in e} \ell_{\mathrm{rec}}\left(\mathbf{x}_v, [A_\psi(\mathcal{H})]_v\right), \tag{5}$$

where A_ψ is an autoencoder trained on the observed hypergraph $\mathcal{H}$, $\mathbf{x}_v$ and $[A_\psi(\mathcal{H})]_v$ denote original input and reconstructed attributes of node v, respectively, and ℓ_{rec} is the reconstruction loss. A larger $a_{\mathrm{H}}(e)$ indicates that the features of the hyperedge's members are harder to reconstruct on average. Further details are provided in Appendix B.

Empirical Analysis. Using the proposed deviation measures, we compare the score distributions of benign and poisoned memberships and hyperedges (on Cora under existing node injection attacks). As shown in Figure 2, poisoned higher-order associations consistently deviate from benign ones, either at the membership level or at the hyperedge level. These distributional gaps provide clear statistical signals for detecting anomalous higher-order associations, highlighting the stealthiness limitations of existing attacks.

4 Methodology

To address the above issues, we propose DPHIA, an effective and unnoticeable node injection attack for hyperedge prediction. As illustrated in Figure 3, DPHIA consists of two stages.(1) **Uncertainty-Guided Poisoning Initialization.** DPHIA constructs an initial poisoned hypergraph by injecting malicious nodes into high-uncertainty hyperedges whose predictions lie near the decision boundary (Section 4.1). (2) **Distribution-Preserving Feature Optimization.** DPHIA optimizes the injected-node features with two objectives, including an adaptive ranking-based objective to mislead hyperedge prediction, and dual-level adversarial regularization to preserve stealthiness (Section 4.2).

4.1 Uncertainty-Guided Poisoning Initialization

During initialization, we first select hyperedges for node injection. Although random injection is simple, it ignores the predictor's decision behavior. To address this, we prioritize high-uncertainty hyperedges near the decision boundary, where small perturbations are more likely to induce label flipping, making them suitable targets for effective yet subtle injection (Wang et al., 2026). Specifically, we first train a predictor f_{θ_c} on the clean $\mathcal{H}$ and keep it fixed during selection. For each $e \in \mathcal{E}$, let $p_e = f_{\theta_c}(\mathcal{H}, e) \in [0, 1]$ denote its predicted probability. We quantify the uncertainty of e as:

$$u(e) = -p_e \log p_e - (1 - p_e) \log(1 - p_e). \tag{6}$$

A larger $u(e)$ indicates higher prediction uncertainty, with the maximum attained at the decision boundary. Given a poisoning rate ρ and a node budget $B = \lfloor \rho |\mathcal{V}| \rfloor$, we select the top-$B$ most

uncertain hyperedges, injecting one malicious node into each (He et al., 2025; 2026). The selected set is denoted by $\mathcal{P} = \{e_i\}_{i=1}^{B}$. To reduce initial feature discrepancies, we then generate the feature of each malicious node using the original members of its selected hyperedge. For each $e_i \in \mathcal{P}$, we obtain the injected-node feature as

$$\mathbf{x}_i^m = g_\omega(\boldsymbol{\mu}_{e_i}), \qquad \boldsymbol{\mu}_{e_i} = \frac{1}{|e_i|} \sum_{v \in e_i} \mathbf{x}_v, \tag{7}$$

where g_ω is a learnable generator whose parameters are optimized in the second stage (Section 4.2). The resulting malicious node v_i^m is then inserted into e_i, yielding $\widehat{e}_i = e_i \cup \{v_i^m\}$.

4.2 Distribution-Preserving Feature Optimization

Once the poisoned hypergraph is initialized, we optimize the generator to refine the injected features, enhancing their ability to mislead the victim model while suppressing distributional deviations.

Attack Effectiveness. To mitigate the attenuation of node-level perturbations in edge predictions, we propose an adaptive ranking-based attack objective that explicitly encourages proxy negatives to outrank proxy positives, thereby weakening edge-level separability and facilitating label flips.

Adaptive Ranking-based Attack Objective. We first initialize an optimization surrogate f_{θ_s} from the clean predictor f_{θ_c} and update it on the poisoned hypergraph $\widehat{\mathcal{H}}$. The observed hyperedges $\mathcal{C}^+ = \widehat{\mathcal{E}}$ serve as proxy positives, while a proxy negative pool $\mathcal{C}^-$ is sampled from unobserved node sets. We then uniformly sample S positive–negative candidate pairs from $\mathcal{C}^+ \times \mathcal{C}^-$ to form the collection $\mathcal{Q}$. For each pair $(c^+, c^-) \in \mathcal{Q}$, let p_{c^+} and p_{c^-} denote the surrogate's predicted probabilities of the positive and negative candidates being hyperedges, respectively. To encourage negative candidates to receive higher probabilities than positives, we optimize the generator by minimizing:

$$\mathcal{L}_{\text{atk}} = \sum_{(c^+,c^-)\in\mathcal{Q}} w(c^+, c^-)\, \sigma\left(\frac{p_{c^+} - p_{c^-}}{\tau}\right), \quad w(c^+, c^-) = \text{softmax}_{\mathcal{Q}}\left(-\frac{|p_{c^+} - p_{c^-}|}{\kappa}\right). \tag{8}$$

where σ is the sigmoid function, $\text{softmax}_{\mathcal{Q}}$ denotes softmax normalization over all pairs in $\mathcal{Q}$, and $\tau, \kappa > 0$ are temperature parameters. The weight $w(c^+, c^-)$ assigns greater importance to pairs with smaller probability gaps, prioritizing their ranking reversal during optimization. By disrupting the relative ordering of predicted probabilities for positive and negative hyperedges, this objective undermines the surrogate's discriminative capacity at the edge level.

Dual-Level Adversarial Regularization. To constrain deviations from the clean data distributions at both the membership and hyperedge levels, we introduce two MLP discriminators trained to distinguish clean from poisoned samples. Specifically, the membership discriminator d_{ϕ_m} assesses local node–hyperedge compatibility, while the hyperedge discriminator d_{ϕ_e} detects distributional shifts in aggregate hyperedge representations. The generator aims to fool both discriminators through the following minimax objectives (Goodfellow et al., 2014):

$$\min_{\omega} \max_{\phi_m} \mathcal{L}_{\text{adv}}^{(m)} = \mathbb{E}_{m\sim P_m^{\text{clean}}}\big[\log d_{\phi_m}(m)\big] + \mathbb{E}_{m\sim P_m^{\omega}}\big[\log(1 - d_{\phi_m}(m))\big], \tag{9}$$

$$\min_{\omega} \max_{\phi_e} \mathcal{L}_{\text{adv}}^{(e)} = \mathbb{E}_{e\sim P_e^{\text{clean}}}\big[\log d_{\phi_e}(e)\big] + \mathbb{E}_{e\sim P_e^{\omega}}\big[\log(1 - d_{\phi_e}(e))\big], \tag{10}$$

where ω denotes the shared generator parameters, and $m = (v, e)$ denotes a node–hyperedge membership. P_m^{clean} and P_e^{clean} denote the distributions for sampling memberships and hyperedges from the clean hypergraph $\mathcal{H}$, while P_m^{ω} and P_e^{ω} denote the corresponding distributions for sampling injected memberships and modified hyperedges from the poisoned hypergraph $\widehat{\mathcal{H}}$, obtained by injecting nodes with features generated by g_ω. Further details on the construction and sampling of membership- and hyperedge-level discriminator inputs are provided in Appendix C.

Joint Optimization. The generator balances attack effectiveness and stealthiness by jointly minimizing the adaptive ranking-based attack loss and the two adversarial objectives defined above. With the surrogate and discriminator parameters fixed, we optimize the generator through

$$\min_{\omega} \quad \mathcal{L}_G = \mathcal{L}_{\text{atk}} + \alpha \mathcal{L}_{\text{adv}}^{(m)} + \beta \mathcal{L}_{\text{adv}}^{(e)}, \tag{11}$$

where $\alpha, \beta \geq 0$ control membership- and hyperedge-level regularization, respectively. Each training round consists of two phases. **(1) Attack optimization.** With the surrogate parameters θ_s fixed, we alternate between two updates: first, we fix the generator and update the discriminators by maximizing $\mathcal{L}_{\text{adv}}^{(m)}$ and $\mathcal{L}_{\text{adv}}^{(e)}$; then, we fix both discriminators and update the generator by minimizing $\mathcal{L}_G$. **(2) Surrogate adaptation.** With the generator fixed, we update the surrogate on the current poisoned hypergraph $\widehat{\mathcal{H}}$. The detailed training algorithm is illustrated in Appendix D.

5 Experiments

In this section, we present a comprehensive empirical evaluation to demonstrate the effectiveness and stealthiness of DPHIA. Additional experimental results on different surrogate backbone, parameter analysis and efficiency are provided in Appendices F.1, F.4 and F.5, respectively.

5.1 Experimental Setup

Datasets. We conduct experiments on five widely used real-world datasets for hyperedge prediction (Hwang et al., 2022; Ko et al., 2025; Yu et al., 2025), including: three co-citation datasets (Citeseer, Cora, and Pubmed), one authorship dataset (DBLP-A), and one collaboration dataset (DBLP). Detailed descriptions and statistics of the datasets are provided in Appendix E.1.

Baselines. We compare it with four representative attack baselines: (1) **Random**, which performs random node injection; (2) Fast Gradient Attack (**FGA**) (Sun et al., 2020), a gradient-based attack adapted to hypergraphs; and two node injection attacks originally developed for hypergraph node classification, (3) **HyperNear** (Cai et al., 2025) and (4) **TH-Attack** (He et al., 2026). To evaluate effectiveness and transferability, all attack methods are conducted in a black-box attack scenario against five widely used hyperedge prediction models: **HNHN** (Dong et al., 2020), **NHP** (Yadati et al., 2020), **AHP** (Hwang et al., 2022), **CASH** (Ko et al., 2025), and **HyGEN** (Yu et al., 2025). Detailed descriptions of each attack method are provided in Appendix E.2.

Evaluation Protocol. Following (Hwang et al., 2022; Ko et al., 2025; Yu et al., 2025), we generate five random splits per dataset, partitioning positive hyperedges into training, validation, and test sets in a 60%:20%:20% ratio. For each split, we train the victim on the poisoned training hypergraph and evaluate its performance on held-out candidates. For validation and testing, we construct four candidate sets using sized negative sampling (SNS), motif negative sampling (MNS), clique negative sampling (CNS), and their mixture (MIX). Each set contains the held-out positive hyperedges and an equal number of negative candidates. As metrics, we use AUC (area under the ROC curve) and AP (average precision), where lower values indicate stronger attacks.

Implementation Details. Following prior works (Cai et al., 2025; He et al., 2026), we set the poisoning rate to $\rho = 5\%$ unless otherwise specified, and vary it from 5% to 30% in the budget analysis in Section 5.4. For DPHIA, both the clean predictor and adaptive surrogate use an HNHN encoder (Dong et al., 2020). More implementation details are provided in Appendix E.3.

5.2 Attack Effectiveness and Transferability

Table 1 reports AUC and AP averaged over different NS strategies for various datasets and victim models (see Appendix F.2 for results under each individual strategy). From the results, we find that **❶ DPHIA consistently outperforms all baselines across the evaluated settings, demonstrating its strong attack effectiveness.** Random, FGA, and TH-Attack show limited attack capability and, in some cases, even improve prediction performance. HyperNear generally ranks second but reduces both metrics by less than 10 points in most cases. In contrast, DPHIA consistently causes substantial degradation in victim performance. For example, on Citeseer, it achieves average reductions of 16.95 points in AUC and 13.13 points in AP across the five victim models. These improvements mainly stem from our ranking-based attack objective that promotes positive–negative ordering reversals at the candidate level, and adaptive weights that prioritize the most influential candidate pairs. **❷ DPHIA remains effective across diverse victims, consistently outperforming state-of-the-art attacks and showing superior cross-model transferability.** For example, on DBLP, DPHIA achieves AUC reductions of 9.57, 8.33, and 20.29 points for NHP, AHP, and HyGEN, respectively,

Table 1: Attack performance across datasets and victim models.

Dataset	Model	Benign		Random		FGA		TH-Attack		HyperNear		DPHIA	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Citeseer	HNHN	72.24	68.25	73.09	69.77	73.66	68.96	70.58	65.61	71.29	67.41	**57.13**	**61.86**
	NHP	70.96	69.70	71.02	68.87	71.09	71.96	70.12	68.89	70.30	69.47	**66.62**	**66.87**
	AHP	76.70	78.27	78.00	79.40	70.01	66.92	81.05	80.69	67.80	69.55	**41.90**	**46.12**
	CASH	77.56	77.73	76.47	78.92	76.15	77.75	78.63	81.34	73.28	74.68	**66.28**	**69.31**
	HyGEN	83.31	83.32	80.76	82.28	75.84	78.18	84.24	84.26	67.39	68.45	**64.07**	**67.45**
Cora	HNHN	64.70	61.73	63.81	61.47	63.02	59.48	63.33	60.29	65.27	62.04	**58.30**	**55.41**
	NHP	60.10	59.14	62.15	63.27	60.74	60.84	60.94	59.65	61.52	62.81	**48.86**	**51.06**
	AHP	70.34	70.69	66.01	67.14	70.90	73.19	72.09	74.19	67.48	67.67	**47.97**	**48.99**
	CASH	72.30	73.90	72.35	73.74	70.72	73.28	70.74	73.41	72.99	72.56	**67.59**	**68.48**
	HyGEN	73.11	68.79	69.02	67.50	68.92	69.27	73.98	70.15	62.82	64.94	**52.11**	**47.92**
Pubmed	HNHN	71.53	69.35	72.96	71.56	73.14	71.65	71.28	67.67	73.97	71.77	**66.97**	**65.68**
	NHP	70.11	69.63	69.83	68.29	69.92	69.36	69.94	69.67	70.73	69.65	**66.07**	**65.50**
	AHP	71.68	70.88	69.99	66.53	65.00	65.00	69.10	68.04	58.52	58.04	**54.02**	**53.38**
	CASH	72.83	73.74	72.53	74.07	69.47	68.84	72.14	73.72	68.13	68.91	**65.85**	**66.33**
	HyGEN	72.48	73.21	69.41	70.74	70.53	72.42	71.00	72.14	70.40	72.37	**67.82**	**67.72**
DBLP-A	HNHN	76.12	74.75	76.69	75.83	76.81	76.17	75.09	72.77	75.97	73.51	**64.26**	**64.83**
	NHP	73.70	73.53	73.58	73.39	74.21	73.54	72.69	72.10	74.74	75.21	**65.67**	**64.42**
	AHP	79.25	82.24	79.46	82.14	79.01	81.74	79.21	82.10	77.71	81.09	**68.74**	**71.33**
	CASH	77.62	81.11	75.35	78.10	76.01	78.75	77.68	81.11	76.16	80.22	**72.09**	**75.01**
	HyGEN	84.23	85.44	84.69	86.18	83.24	85.37	84.20	85.84	80.09	83.35	**73.05**	**70.94**
DBLP	HNHN	76.70	77.12	72.06	72.10	72.07	71.93	72.10	72.18	72.04	71.77	**52.65**	**52.55**
	NHP	58.22	60.65	58.22	60.80	58.24	60.70	58.43	60.85	58.47	60.80	**48.65**	**46.01**
	AHP	73.69	72.88	73.68	72.96	73.60	72.89	73.09	72.84	73.59	72.81	**65.36**	**66.23**
	CASH	73.09	72.58	72.76	71.48	71.47	72.57	70.19	70.91	70.65	71.32	**68.65**	**68.42**
	HyGEN	74.04	73.06	71.90	71.85	72.01	71.76	71.97	71.67	72.05	71.77	**53.75**	**48.66**

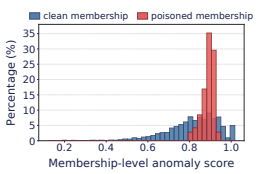
Table 2: Attack performance against HyGEN under different defenses.

Dataset	Benign		Defense	Random		FGA		TH-Attack		HyperNear		DPHIA	
	AUC	AP		AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Citeseer	83.31	83.32	None	80.76	82.28	75.84	78.18	84.24	84.26	67.39	68.45	**64.07**	**67.45**
			MD-filtering	83.58	81.73	80.68	78.89	83.18	83.78	73.04	72.91	**64.34**	**67.23**
			HD-filtering	82.26	82.28	81.19	81.68	83.47	83.44	76.78	77.48	**63.28**	**66.28**
Cora	73.11	68.79	None	69.02	67.50	68.92	69.27	73.98	70.15	62.82	64.94	**52.11**	**47.92**
			MD-filtering	74.74	74.94	72.94	72.87	74.02	70.81	68.85	69.38	**55.69**	**58.39**
			HD-filtering	74.94	73.80	74.31	73.02	72.91	70.05	70.94	71.60	**52.90**	**48.89**
Pubmed	72.48	73.21	None	69.41	70.74	70.53	72.42	71.00	72.14	70.40	72.37	**67.82**	**67.72**
			MD-filtering	69.51	70.91	71.82	73.93	70.78	71.79	73.64	72.73	**68.01**	**69.04**
			HD-filtering	71.04	71.99	71.72	73.30	70.64	70.87	72.25	73.67	**66.87**	**66.76**
DBLP-A	84.23	85.44	None	84.69	86.18	83.24	85.37	84.20	85.84	80.09	83.35	**73.05**	**70.94**
			MD-filtering	82.09	82.49	80.26	79.06	84.70	85.82	81.24	82.91	**72.70**	**70.66**
			HD-filtering	84.14	85.30	83.51	85.21	84.06	84.65	83.05	83.67	**73.07**	**71.87**
DBLP	74.04	73.06	None	71.90	71.85	72.01	71.76	71.97	71.67	72.05	71.77	**53.75**	**48.66**
			MD-filtering	71.99	72.09	71.60	71.56	73.63	72.86	72.73	72.17	**57.13**	**57.41**
			HD-filtering	74.69	73.18	72.06	72.23	72.68	72.15	73.18	73.52	**57.87**	**58.40**

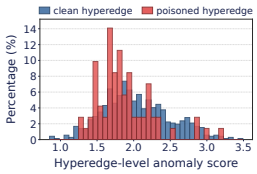
while HyperNear has little impact on their prediction performance. This robustness largely benefits from our architecture-agnostic adaptive ranking objective, which optimizes candidate ordering rather than model-specific internal representations.

5.3 Stealthiness Analysis

To evaluate DPHIA's stealthiness, we utilize the membership- and hyperedge-level deviation scores defined in Section 3.2. We first examine the anomaly-score distributions. As shown in Figure 4, the clean and poisoned samples exhibit highly overlapping score distributions on Cora, indicating limited distinguishability under both metrics. We then construct two filtering defenses, MD-filtering and HD-filtering, to remove memberships and hyperedges with the highest deviation scores, respectively. We set the filtering ratio to 5% for both defenses and reevaluate attack effectiveness on HyGEN after filtering. As shown in Table 2, DPHIA consistently maintains the strongest attack performance across datasets. For example, HD-filtering substantially weakens HyperNear on Citeseer, increasing the victim AUC from 67.39% to 76.78%. In contrast, DPHIA retains AUC values of 64.34% under MD-filtering and 63.28% under HD-filtering, both close to 64.07% without filtering. This is attributed to our dual-level adversarial regularization, which effectively reduces detectable distributional deviations while preserving attack effectiveness.

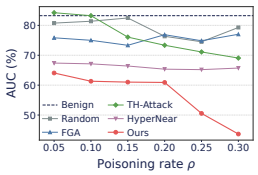


(a) DPHIA (MD).

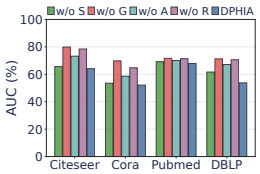


(b) DPHIA (HD).

Figure 4: Anomaly-score distributions under membership- and hyperedge-level detection.



(a) Varying poisoning rate.



(b) Ablation study.

Figure 5: Effects of the poisoning rate and individual components on attack performance.

5.4 Impact of the Attack Budget

To assess the impact of attack budget, we vary the poisoning rate ρ from 5% to 30% on Citeseer using HyGEN as the victim. As shown in Figure 5a, DPHIA achieves substantial prediction degradation even at low poisoning rates and continues to improve as the budget increases. In contrast, Random and FGA induce only limited performance degradation and remain relatively insensitive to changes in the poisoning rate. HyperNear shows only limited gains with additional injections, while TH-Attack becomes competitive only at higher rates. These results indicate that increasing the injection budget alone does not guarantee stronger attacks. By explicitly promoting candidate ranking reversals, DPHIA benefits more consistently from additional injections.

5.5 Ablation Study

To assess the contribution of each component in DPHIA, we evaluate four ablation variants against HyGEN. Specifically, **w/o S** replaces uncertainty-guided target selection with random selection; **w/o G** removes generator optimization; **w/o A** removes the adaptive pair weight; and **w/o R** replaces the ranking-based attack objective with binary cross-entropy maximization. As shown in Figure 5b, removing any component results in performance degradation, underscoring the necessity of each design in the attack framework. **w/o G** and **w/o R** cause the largest degradation across most datasets, highlighting the critical roles of injected-node feature optimization and ranking-based attack optimization. The notable degradation of **w/o A** demonstrates the contribution of adaptive pair weighting to attack effectiveness. The impact of adversarial regularization is examined in Appendix F.3.

6 Related Works

6.1 Adversarial Attacks on Hypergraphs

Adversarial attacks have emerged as an important research direction in hypergraph learning (Hu et al., 2023; Chen et al., 2023; Arafat et al., 2026). Among these, node injection offers a practical and potentially stealthy attack strategy. Specifically, H3NI (Shi et al., 2025) employs genetic search to determine injection locations, whereas IE-Attack (He et al., 2025) selects elite hyperedges and inserts nodes with homogeneous features. HyperNear (Cai et al., 2025) emphasizes stealthiness by preserving homophily during attack optimization. TH-Attack (He et al., 2026) injects semantically divergent nodes into pivotal hyperedges to improve transferability. Despite these advances, existing methods primarily target node classification, leaving node injection attacks on hyperedge prediction largely unexplored. We address this gap with DPHIA, the first node injection attack specifically designed for hyperedge prediction. Additional related work is discussed in Appendix G.

7 Conclusion

We systematically investigated the vulnerability of hyperedge prediction to node injection poisoning. Our theoretical analysis shows that aggregating node embeddings into hyperedge representations can attenuate perturbation effects, limiting attack effectiveness. Our empirical analysis further reveals detectable deviations at the membership and hyperedge levels after injection, compromising attack stealthiness. To address these challenges, we proposed DPHIA, a node injection framework that combines uncertainty-guided injection target selection with distribution-preserving feature optimization. Extensive experiments demonstrate its superior attack utility and low detectability.

AI USE STATEMENT

We used generative AI tools for proofreading, minor linguistic refinement, and code review. We have reviewed all AI-assisted content and take full responsibility for the final manuscript and code.

ETHICS STATEMENT

This work advances the study of adversarial robustness for decision-making systems that rely on higher-order relational modeling. By exposing vulnerabilities of hyperedge prediction to node injection attacks, we aim to support robustness evaluation and the development of effective defenses. Our experimental study does not raise specific ethical concerns. All datasets are publicly available and have been released for academic purposes, containing no personally identifiable information or offensive content. All experiments are conducted in a controlled research setting, without targeting deployed systems. Given the potential for misuse, we encourage using the proposed method only in authorized settings and reporting robustness results transparently.

REPRODUCIBILITY STATEMENT

We describe DPHIA and its optimization procedure in Section 4. The datasets, data splits, evaluation protocol, and implementation details are provided in Section 5.1 and Appendix E.3. Proofs of our theoretical results are included in Appendix A. Our source code and configuration files are available at `https://anonymous.4open.science/r/DPHIA-063D/`.

REFERENCES

Naheed Anjum Arafat, Debabrota Basu, Yulia Gel, and Danda B. Rawat. Adversarial attacks and robust training for hypergraph neural networks. In *Forty-third International Conference on Machine Learning*, 2026. URL `https://openreview.net/forum?id=9NPxvxmZ7e`.

Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021.

Ali Behrouz, Farnoosh Hashemi, Sadaf Sadeghian, and Margo Seltzer. Cat-walk: Inductive hypergraph learning via set walks. *Advances in Neural Information Processing Systems*, 36:32636–32671, 2023.

Tingyi Cai, Yunliang Jiang, Ming Li, Lu Bai, Changqin Huang, and Yi Wang. Hypernear: Unnoticeable node injection attacks on hypergraph neural networks. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=Toy3nwPhk4`.

Cen Chen, Chen Liang, Jianbin Lin, Li Wang, Ziqi Liu, Xinxing Yang, Jun Zhou, Yang Shuang, and Yuan Qi. Infdetect: A large scale graph-based fraud detection system for e-commerce insurance. In *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1765–1773. IEEE, 2019.

Yang Chen, Stjepan Picek, Zhonglin Ye, Zhaoyang Wang, and Haixing Zhao. Momentum gradient-based untargeted attack on hypergraph neural networks. *arXiv preprint arXiv:2310.15656*, 2023.

Yongqiang Chen, Han Yang, Yonggang Zhang, Ma Kaili, Tongliang Liu, Bo Han, and James Cheng. Understanding and improving graph injection attack by promoting unnoticeability. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=wkMG8cdvh7-`.

Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. *arXiv preprint arXiv:2106.13264*, 2021.

Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM international conference on data mining*, pp. 594–602. SIAM, 2019.

Yihe Dong, Will Sawin, and Yoshua Bengio. Hnhn: Hypergraph networks with hyperedge neurons. *arXiv preprint arXiv:2006.12278*, 2020.

Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 3558–3565, 2019.

Yifan Feng, Yifan Zhang, Shaoyi Du, Shihui Ying, Jun-Hai Yong, and Yue Gao. Hgnn shield: Defending hypergraph neural networks against high-order structure attack. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Meixia He, Peican Zhu, Keke Tang, and Yangming Guo. Hypergraph attacks via injecting homogeneous nodes into elite hyperedges. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 282–290, 2025.

Meixia He, Peican Zhu, Le Cheng, Yangming Guo, Manman Yuan, and Keke Tang. Transferable hypergraph attack via injecting nodes into pivotal hyperedges. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 372–380, 2026.

Chao Hu, Ruishi Yu, Binqi Zeng, Yu Zhan, Ying Fu, Quan Zhang, Rongkai Liu, and Heyuan Shi. Hyperattack: Multi-gradient-guided white-box adversarial structure attack of hypergraph neural networks. *arXiv preprint arXiv:2302.12407*, 2023.

Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. *arXiv preprint arXiv:2105.00956*, 2021.

Hyunjin Hwang, Seungwoo Lee, Chanyoung Park, and Kijung Shin. Ahp: Learning to negative sample for hyperedge prediction. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pp. 2237–2242, 2022.

Sunwoo Kim, Soo Yong Lee, Yue Gao, Alessia Antelmi, Mirko Polato, and Kijung Shin. A survey on hypergraph neural networks: An in-depth and step-by-step guide. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6534–6544, 2024.

Yunyong Ko, Hanghang Tong, and Sang-Wook Kim. Enhancing hyperedge prediction with context-aware self-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 37(4): 1772–1784, 2025.

Fan Li, Xiaoyang Wang, Wenjie Zhang, Ying Zhang, and Xuemin Lin. Dhg-bench: A comprehensive benchmark for deep hypergraph learning. In *International Conference on Learning Representations*, volume 2026, pp. 47693–47727, 2026a.

Ming Li, Huiting Wang, Yuting Chen, Lu Bai, Lixin Cui, Feilong Cao, and Ke Lv. Self-supervised hypergraph learning with substructure awareness for hyperedge prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23055–23062, 2026b.

Ming Li, Zhanle Zhu, Xinyi Li, Lu Bai, Lixin Cui, Feilong Cao, and Ke Lv. Hypernora: Hyperedge prediction via node-level relation-aware self-supervised hypergraph learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23047–23054, 2026c.

Rui Ou, Kun Zhu, Nana Zhang, Jiangtong Li, Chaochao Chen, Yuhua Xu, and Changjun Jiang. Targeting borderline fraudsters: multi-view hypergraph fraud detection with llm-guided contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 15591–15599, 2026.

Khaled Mohammed Saifuddin, Briana Bumgardner, Farhan Tanvir, and Esra Akbas. Hygnn: Drug-drug interaction prediction via hypergraph neural network. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pp. 1503–1516. IEEE, 2023.

Heyuan Shi, Binqi Zeng, Ruishi Yu, Yulin Yang, Zijian Zouxia, Chao Hu, and Ronghua Shi. H3ni: Non-target-specific node injection attacks on hypergraph neural networks via genetic algorithm. *Neurocomputing*, 613:128746, 2025.

Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of the web conference 2020*, pp. 673–683, 2020.

Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. Equivariant hypergraph diffusion neural operators. *arXiv preprint arXiv:2207.06680*, 2022.

Xiaobao Wang, Ruoxiao Sun, Yujun Zhang, Bingdao Feng, Dongxiao He, Luzhi Wang, and Di Jin. Stealthy yet effective: Distribution-preserving backdoor attacks on graph classification. *Advances in Neural Information Processing Systems*, 38:65501–65529, 2026.

Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. Self-supervised hypergraph convolutional networks for session-based recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 4503–4511, 2021.

Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. Nhp: Neural hypergraph link prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, pp. 1705–1714, 2020.

Se-eun Yoon, Hyungseok Song, Kijung Shin, and Yung Yi. How much and when do we need higher-order information in hypergraphs? a case study on hyperedge prediction. In *Proceedings of the web conference 2020*, pp. 2627–2633, 2020.

Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proceedings of the web conference 2021*, pp. 413–424, 2021.

Song Kyung Yu, Da Eun Lee, Yunyong Ko, and Sang-Wook Kim. Hygen: Regularizing negative hyperedge generation for accurate hyperedge prediction. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 1500–1504, 2025.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. *arXiv preprint arXiv:1911.02613*, 2019.

APPENDIX

A THEORETICAL ANALYSIS

Setup. We follow the notation in Section 3.1. Let $\mathbf{z}_v$ and $\widehat{\mathbf{z}}_v$ denote node embeddings before and after injection, respectively. For candidate c, $\mathbf{q}_c$ and $\widehat{\mathbf{q}}_c$ are the corresponding max–min representations. The embedding perturbation is $\boldsymbol{\delta}_v = \widehat{\mathbf{z}}_v - \mathbf{z}_v$, with mean $\mathbf{a}_c = |c|^{-1} \sum_{v \in c} \boldsymbol{\delta}_v$ and maximum deviation $\varepsilon_c = \max_{v \in c} \|\boldsymbol{\delta}_v - \mathbf{a}_c\|_\infty$ within c.

A.1 PROOF OF LEMMA 1.

Lemma 1 (Perturbation attenuation under candidate aggregation)**.** *For any candidate c, the shift in its aggregated representation is bounded by*

$$\|\widehat{\mathbf{q}}_c - \mathbf{q}_c\|_\infty \leq 2\varepsilon_c. \tag{2}$$

Proof. For a candidate c, consider the j-th dimension of its member node embeddings. By definition, ε_c bounds how much each node's perturbation differs from the mean, so $|\delta_{v,j} - a_{c,j}| \leq \varepsilon_c$ for every $v \in c$. Since $\widehat{z}_{v,j} = z_{v,j} + \delta_{v,j}$, we obtain

$$z_{v,j} + a_{c,j} - \varepsilon_c \leq \widehat{z}_{v,j} \leq z_{v,j} + a_{c,j} + \varepsilon_c. \tag{12}$$

Thus, each perturbed value lies within ε_c of its original value shifted by $a_{c,j}$. Taking the maximum over members gives the same bound for the largest value, and applying the same argument to the minimum gives

$$\begin{aligned} \left| \max_{v \in c} \widehat{z}_{v,j} - \max_{v \in c} z_{v,j} - a_{c,j} \right| &\leq \varepsilon_c, \\ \left| \min_{v \in c} \widehat{z}_{v,j} - \min_{v \in c} z_{v,j} - a_{c,j} \right| &\leq \varepsilon_c. \end{aligned} \tag{13}$$

Max–min pooling subtracts the minimum from the maximum. The mean shift $a_{c,j}$ therefore cancels, leaving only the deviations of these two values from that shift. Each deviation is at most ε_c, so the triangle inequality gives

$$|[\widehat{\mathbf{q}}_c]_j - [\mathbf{q}_c]_j| \leq 2\varepsilon_c. \tag{14}$$

Since this holds for every dimension j, taking the maximum over dimensions proves Eq. 2. In particular, when $\varepsilon_c = 0$, all member embeddings change by the same vector $\mathbf{a}_c$. Their max–min representation therefore remains unchanged, even if this shared change is large. □

A.2 PROOF OF COROLLARY 1

Corollary 1 (Ranking stability under perturbation attenuation)**.** *Consider a positive–negative candidate pair (c^+, c^-) and a fixed prediction head g that maps candidate representations to hyperedge probabilities. Assume that g is L_a-Lipschitz with respect to $\|\cdot\|_\infty$, where $L_a > 0$. Let $p_c = g(\mathbf{q}_c)$ and $\widehat{p}_c = g(\widehat{\mathbf{q}}_c)$ denote the predicted hyperedge probabilities of candidate c before and after injection, respectively. For a pair correctly predicted before injection, the initial probability gap is $\Delta = p_{c^+} - p_{c^-} > 0$. To disrupt this ranking, the attacker seeks $\widehat{\Delta} = \widehat{p}_{c^+} - \widehat{p}_{c^-} \leq 0$, which requires*

$$\varepsilon_{c^+} + \varepsilon_{c^-} \geq \frac{\Delta}{2L_a}. \tag{3}$$

Thus, a larger initial gap requires greater differences among node perturbations within the two candidates. If this condition is not met, the positive candidate remains ranked above the negative.

Proof. By the Lipschitz condition on the fixed prediction head g and Lemma 1, the probability change for each candidate c is bounded by

$$\begin{aligned} |\widehat{p}_c - p_c| &= |g(\widehat{\mathbf{q}}_c) - g(\mathbf{q}_c)| \\ &\leq L_a \|\widehat{\mathbf{q}}_c - \mathbf{q}_c\|_\infty \\ &\leq 2L_a \varepsilon_c. \end{aligned} \tag{15}$$

For an initially correctly ranked pair (c^+, c^-), let $\Delta = p_{c^+} - p_{c^-} > 0$ and $\widehat{\Delta} = \widehat{p}_{c^+} - \widehat{p}_{c^-}$ denote the probability gaps before and after injection, respectively. Applying the triangle inequality and Eq. 15 yields

$$\begin{aligned} |\widehat{\Delta} - \Delta| &= |(\widehat{p}_{c^+} - p_{c^+}) - (\widehat{p}_{c^-} - p_{c^-})| \\ &\leq |\widehat{p}_{c^+} - p_{c^+}| + |\widehat{p}_{c^-} - p_{c^-}| \\ &\leq 2L_a(\varepsilon_{c^+} + \varepsilon_{c^-}). \end{aligned} \tag{16}$$

To make the negative candidate's predicted probability equal to or higher than the positive candidate's after injection, the attacker must achieve $\widehat{\Delta} \leq 0$. The probability gap must therefore decrease by at least Δ. Together with Eq. 16, this gives

$$\Delta \leq \Delta - \widehat{\Delta} = |\widehat{\Delta} - \Delta| \leq 2L_a(\varepsilon_{c^+} + \varepsilon_{c^-}). \tag{17}$$

Dividing both sides by $2L_a > 0$ yields

$$\varepsilon_{c^+} + \varepsilon_{c^-} \geq \frac{\Delta}{2L_a}. \tag{3}$$

This establishes the necessary condition for achieving $\widehat{\Delta} \leq 0$. If this condition is not satisfied, the positive candidate remains ranked above the negative. □

B Hyperedge-level Deviation Metric

We detail the computation of the hyperedge-level deviation score defined in Eq. 5.

Autoencoder. Inspired by reconstruction-based outlier detection (Ding et al., 2019), we use an autoencoder A_ψ comprising an HNN encoder and an MLP decoder. The encoder learns node representations from attributes and hypergraph structure, while the decoder reconstructs node attributes. Given the observed hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, the reconstructed attribute vector of node v is

$$\widehat{\mathbf{x}}_v = [A_\psi(\mathcal{H})]_v, \tag{18}$$

where ψ contains the parameters of both the encoder and decoder, and $[\cdot]_v$ selects the output corresponding to node v.

Reconstruction objective. We measure reconstruction error using the squared L_2 distance:

$$\ell_{\text{rec}}(\mathbf{x}_v, \widehat{\mathbf{x}}_v) = \|\mathbf{x}_v - \widehat{\mathbf{x}}_v\|_2^2. \tag{19}$$

The autoencoder is trained by minimizing the average reconstruction error over all observed nodes:

$$\mathcal{L}_{\text{rec}}(\psi) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \ell_{\text{rec}}\big(\mathbf{x}_v, [A_\psi(\mathcal{H})]_v\big). \tag{20}$$

Hyperedge-level deviation score. After training, we freeze A_ψ and compute each node's reconstruction error. For each hyperedge e, we average the reconstruction errors of its members:

$$a_{\text{H}}(e) = \frac{1}{|e|} \sum_{v \in e} \ell_{\text{rec}}(\mathbf{x}_v, \widehat{\mathbf{x}}_v), \tag{21}$$

A larger score indicates that the attributes of the hyperedge's members are harder to reconstruct from patterns learned over benign data.

C Discriminator Input Construction and Sampling

We detail how membership and hyperedge samples are represented and sampled for the adversarial objectives in Eqs. 9 and 10.

Membership Discriminator Inputs. For a membership $m = (v, e)$, the discriminator d_{ϕ_m} input concatenates the node feature, the mean feature of the other members, and their difference:

$$\mathbf{u}_m(v, e) = \left[\mathbf{x}_v \parallel \boldsymbol{\mu}_{e\setminus\{v\}} \parallel \left(\mathbf{x}_v - \boldsymbol{\mu}_{e\setminus\{v\}}\right)\right], \tag{22}$$

Algorithm 1 DPHIA: Distribution-Preserving Hypergraph Injection Attack

Input: Observed hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$; poisoning rate ρ; optimization rounds R; attack-update steps M; surrogate-update steps T; candidate-pair sampling size S; loss parameters $\tau, \kappa, \alpha, \beta$; learning rates $\eta_g, \eta_s, \eta_m, \eta_e$.

Output: Poisoned hypergraph $\widehat{\mathcal{H}}$.

1: **Stage I: Uncertainty-guided Poisoned Hypergraph Initialization**
2: Set the node budget $B \leftarrow \lfloor \rho |\mathcal{V}| \rfloor$.
3: Train the clean predictor f_{θ_c} on $\mathcal{H}$.
4: Compute $u(e)$ for each $e \in \mathcal{E}$ using Eq. 6.
5: Select the top-B hyperedges as $\mathcal{P} = \{e_i\}_{i=1}^{B}$.
6: Initialize the generator g_ω.
7: For each $e_i \in \mathcal{P}$, compute $\boldsymbol{\mu}_i$, generate $\mathbf{x}_i^m = g_\omega(\boldsymbol{\mu}_i)$, and inject v_i^m into e_i to construct $\widehat{\mathcal{H}}$.
8: **Stage II: Distribution-Preserving Feature Optimization**
9: Initialize $\theta_s \leftarrow \theta_c$ and discriminator parameters ϕ_m, ϕ_e.
10: Set $\mathcal{C}^+ \leftarrow \widehat{\mathcal{E}}$ and construct the proxy negative pool $\mathcal{C}^-$ from unobserved node sets.
11: Sample S candidate pairs $\mathcal{Q}$ uniformly from $\mathcal{C}^+ \times \mathcal{C}^-$.
12: **for** $r = 1, \ldots, R$ **do**
(1) Attack optimization
13: **for** $j = 1, \ldots, M$ **do**
14: Update ϕ_m and ϕ_e by maximizing $\mathcal{L}_{\text{adv}}^{(m)}$ and $\mathcal{L}_{\text{adv}}^{(e)}$, respectively, with ω and θ_s fixed.
15: Update ω by minimizing $\mathcal{L}_G$ using Eq. 11, with θ_s, ϕ_m, ϕ_e fixed.
16: Regenerate the injected features and update $\widehat{\mathcal{H}}$.
17: **end for**
(2) Surrogate adaptation
18: **for** $t = 1, \ldots, T$ **do**
19: Update θ_s by minimizing the prediction loss over $\mathcal{C}^+$ and $\mathcal{C}^-$ on $\widehat{\mathcal{H}}$, with ω fixed.
20: **end for**
21: **end for**
22: **return** $\widehat{\mathcal{H}}$.

where $\boldsymbol{\mu}_{e\setminus\{v\}}$ is defined in Eq. 4, and $\|$ denotes concatenation. Clean samples from P_m^{clean} are original memberships (v, e) with $e \in \mathcal{E}$ and $v \in e$, while poisoned samples from P_m^{ω} are injected memberships $(v_i^m, \widehat{e}_i)$ with $e_i \in \mathcal{P}$.

Hyperedge Discriminator Inputs. For a hyperedge e, the discriminator d_{ϕ_e} input concatenates its mean member feature and its difference from a clean reference:

$$\mathbf{u}_e(e) = \left[\boldsymbol{\mu}_e \,\|\, (\boldsymbol{\mu}_e - \boldsymbol{\mu}_{\text{clean}})\right], \tag{23}$$

where $\boldsymbol{\mu}_e = |e|^{-1} \sum_{v \in e} \mathbf{x}_v$ and $\boldsymbol{\mu}_{\text{clean}} = |\mathcal{E}|^{-1} \sum_{e' \in \mathcal{E}} \boldsymbol{\mu}_{e'}$. Clean samples from P_e^{clean} are hyperedges $e \in \mathcal{E}$, while poisoned samples from P_e^{ω} are modified hyperedges $\widehat{e}_i$ with $e_i \in \mathcal{P}$.

D ALGORITHM

Algorithm 1 summarizes DPHIA. Stage I (lines 1–7) trains the clean predictor, selects high-uncertainty hyperedges, and initializes the poisoned hypergraph by injecting generated nodes. Stage II initializes the adaptive surrogate, discriminators, proxy candidate sets, and candidate pairs (lines 8–11). It then alternates between **attack optimization** (lines 13–17) and **surrogate adaptation** (lines 18–20). Attack optimization alternately updates the discriminators and generator to balance attack effectiveness and stealthiness. Surrogate adaptation updates the surrogate on the current poisoned hypergraph. The final poisoned hypergraph is returned in line 22.

Table 3: Dataset statistics.

Datasets	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	# Features	Type
Citeseer	1,457	1,078	3,703	Co-citation
Cora	1,434	1,579	1,433	Co-citation
Pubmed	3,840	7,962	500	Co-citation
DBLP-A	39,283	16,483	4,543	Authorship
DBLP	15,639	22,964	4,543	Collaboration

E Additional Experimental Setups

E.1 Datasets

We use five real-world hypergraph benchmarks covering co-citation, authorship, and collaboration relations (Hwang et al., 2022; Ko et al., 2025; Yu et al., 2025). Table 3 summarizes their statistics. The hypergraph representations and node features are described below.

Co-citation datasets. Citeseer, Cora, and Pubmed[1] represent citation relationships among scientific papers. Each node corresponds to a paper, and each hyperedge contains the papers cited by a given paper. Node features are bag-of-words representations of paper abstracts.

Authorship datasets. DBLP-A[2] represent relationships among papers through shared authorship. Each node corresponds to a paper, and each hyperedge contains the papers written or coauthored by a given researcher. Node features are bag-of-words representations of paper abstracts.

Collaboration dataset. DBLP represents collaboration relationships among researchers, using publication records from 87 venues in the AMiner academic network[3]. Each node corresponds to a researcher, and each hyperedge contains the authors of a paper. Node features are obtained by averaging the bag-of-words representations of the abstracts of each researcher's papers.

E.2 Baselines

All attacks follow the black-box poisoning setting in Section 2, with access only to the observed training hypergraph. For fairness, they inject malicious nodes into existing hyperedges under the same budget $B = \lfloor \rho|\mathcal{V}| \rfloor$, while preserving original features and incidences.

Random. Random selects host hyperedges at random and injects one malicious node per host, with node features sampled from a Gaussian distribution.

Fast Gradient Attack (FGA). FGA (Sun et al., 2020) uses gradients to guide the connections of injected nodes. Following TH-Attack (He et al., 2026), we adapt it to hypergraphs by selecting host hyperedges with the largest absolute gradient scores, computed from the hyperedge prediction loss using the shared surrogate. One malicious node is injected per selected host.

HyperNear. HyperNear (Cai et al., 2025) randomly selects host hyperedges and preserves feature-based homophily to improve stealthiness. We replace its node-classification loss with binary cross-entropy over hyperedge candidates, retaining the homophily regularizer and computing attack gradients through the shared surrogate without querying the victim. Following the original configuration, we inject five nodes per selected host. To maintain a comparable perturbation budget, we omit hyperedge splitting and structural updates that modify original incidences.

TH-Attack. TH-Attack (He et al., 2026) identifies pivotal host hyperedges from the observed structure and generates malicious nodes through a feature inverter that promotes semantic divergence from their hosts. We retain its model-independent feature-inversion objective, which requires neither node-classification labels nor victim gradients. One malicious node is injected per selected host before victim training, and attack effectiveness is evaluated on hyperedge prediction.

[1] https://linqs.soe.ucsc.edu/data

[2] https://github.com/malllabiisc/HyperGCN

[3] https://lfs.aminer.cn/lab-datasets/citation/DBLP-citation-network-Oct-19.tar.gz

Table 4: Attack performance against HyGEN using different surrogate hypergraph encoders.

Dataset	Encoder	AUC	AP	Dataset	Encoder	AUC	AP
Citeseer	HGNN	69.23	69.69	Cora	HGNN	59.37	55.27
	HNHN	**64.07**	**67.45**		HNHN	**52.11**	**47.92**
	HCHA	70.19	72.25		HCHA	55.89	58.16
	AllSetFormer	79.68	80.72		AllSetFormer	60.70	62.75
	UniGIN	68.47	70.41		UniGIN	59.33	60.64
	ED-HNN	74.31	74.57		ED-HNN	61.32	63.18

E.3 Implementation Details.

All experiments are run on a single NVIDIA RTX A5000 GPU with 24 GB of memory, using Python 3.9.23, PyTorch 2.4.0, and CUDA 12.1. Baseline implementations follow the authors' official code. For DPHIA, both the clean predictor and the adaptive surrogate use an HNHN encoder (Dong et al., 2020) followed by an MLP prediction head with two hidden layers. The generator is a residual MLP, while both discriminators are MLPs with one hidden layer. Optimization uses $R = 5$ alternating rounds, with $M = 20$ generator updates and $T = 20$ surrogate updates per round. The candidate-pair sampling size is $S = 512$ for Citeseer, Cora, and Pubmed, and $S = 2{,}048$ for DBLP and DBLP-A. DPHIA uses MIX negative sampling, drawing approximately equal proportions from SNS, MNS, and CNS. The poisoning rate is $\rho = 5\%$ in the main experiments and ranges from 5% to 30% in the budget analysis. The regularization weights α and β are tuned over $\{0, 0.25, 0.5, 0.75, 1.0\}$ separately for the undefended, MD-filtering, and HD-filtering settings.

F Additional Experimental Results

F.1 Impact of Different Surrogate Backbones

An effective surrogate should be expressive enough to guide attack optimization and enable the resulting perturbations to transfer to unknown victim architectures. To examine DPHIA's sensitivity to surrogate choice, we consider six surrogate hypergraph encoders: HGNN (Feng et al., 2019), HNHN (Dong et al., 2020), HCHA (Bai et al., 2021), AllSetFormer (Chien et al., 2021), UniGIN (Huang & Yang, 2021), and ED-HNN (Wang et al., 2022). These encoders provide architectural diversity for evaluating whether DPHIA remains effective when the surrogate architecture changes. We conduct experiments on Citeseer and Cora with HyGEN as the victim. Table 4 reports the attack performance. All six surrogate encoders yield lower AUC and AP than the benign results in Table 1 on both datasets, indicating that DPHIA's effectiveness is not restricted to a specific surrogate architecture. However, attack strength varies across encoders. For example, on Citeseer, HNHN and AllSetFormer reduce victim AUC by 19.24 and 3.63 points, respectively. HNHN consistently achieves the lowest AUC and AP on both datasets, supporting its selection as the default surrogate encoder. These results show that DPHIA remains effective across different surrogates, while the choice of surrogate influences the strength of the transferred attacks.

F.2 Impact of Different Negative Sampling Strategies

In most realistic attack scenarios, the attacker has no access to the victim's negative sampling strategy. We therefore evaluate DPHIA against HyGEN under SNS, MNS, CNS, and MIX across five datasets, while fixing the attacker's proxy sampling strategy to MIX. As shown in Table 5, DPHIA reduces both AUC and AP in almost all settings, demonstrating strong transferability across negative sampling strategies. CNS presents the most challenging prediction setting, as its densely connected negatives structurally resemble positive hyperedges, resulting in the lowest benign performance. Conversely, SNS generates random negatives that are typically less densely connected and easier to distinguish, yielding the highest benign performance. Nevertheless, the substantial degradation under SNS shows that high benign performance does not necessarily imply robustness to poisoning, highlighting the importance of evaluating robustness across diverse negative sampling strategies.

Table 5: Attack performance against HyGEN under different negative sampling strategies. Bold indicates a decrease relative to the corresponding benign setting.

Dataset	Setting	SNS		MNS		CNS		MIX		AVG	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Citeseer	Benign	96.53	97.39	86.89	86.14	66.30	67.85	83.52	81.89	83.31	83.32
	Ours	**67.14**	**71.45**	**68.17**	**69.45**	**57.73**	**59.44**	**63.22**	**69.46**	**64.07**	**67.45**
Cora	Benign	91.88	90.57	78.39	72.94	51.05	48.90	71.11	62.75	73.11	68.79
	Ours	**51.59**	**45.86**	**48.38**	**45.01**	55.10	52.73	**53.35**	**48.07**	**52.11**	**47.92**
Pubmed	Benign	89.06	88.94	78.99	79.37	49.38	51.31	72.48	73.21	72.48	73.21
	Ours	**77.30**	**77.40**	**77.79**	**78.36**	**48.28**	**49.77**	**67.92**	**65.34**	**67.82**	**67.72**
DBLP-A	Benign	96.35	96.96	91.25	91.88	65.12	69.19	84.21	83.71	84.23	85.44
	Ours	**87.22**	**87.57**	**82.41**	**80.55**	**49.23**	**48.44**	**73.33**	**67.20**	**73.05**	**70.94**
DBLP	Benign	88.74	89.70	76.77	76.22	56.50	55.58	74.14	70.75	74.04	73.06
	Ours	**58.77**	**49.88**	**51.31**	**46.89**	**51.19**	**49.46**	**53.74**	**48.41**	**53.75**	**48.66**

Table 6: Effect of adversarial regularization on attack performance under different defenses. Bold indicates better attack performance.

Dataset	Benign		Defense	w/o Regularization		DPHIA	
	AUC	AP		AUC	AP	AUC	AP
Citeseer	83.31	83.32	None	**62.13**	**63.28**	64.07	67.45
			MD-filtering	76.52	74.36	**64.34**	**67.23**
			HD-filtering	73.52	72.05	**63.28**	**66.28**
Cora	73.11	68.79	None	**50.08**	**46.64**	52.11	47.92
			MD-filtering	68.47	70.24	**55.69**	**58.39**
			HD-filtering	65.20	64.55	**52.90**	**48.89**

F.3 Impact of adversarial regularization

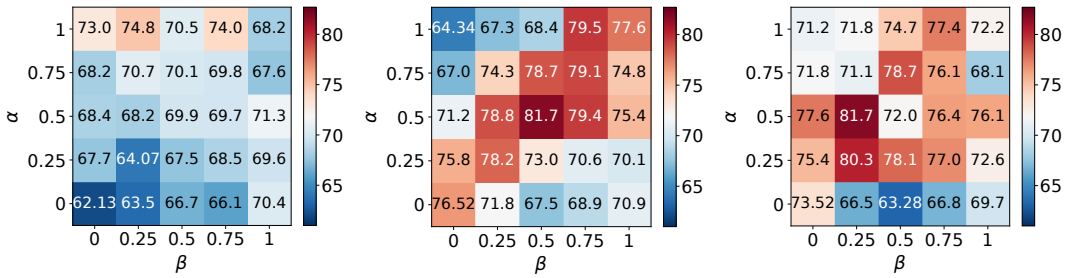
To assess the contribution of adversarial training, we compare DPHIA with **w/o Regularization**, which removes both adversarial regularization terms. We conduct experiments on Citeseer and Cora using HyGEN as the victim model, evaluating attack performance without defense and under the MD-filtering and HD-filtering defenses introduced in Section 5.3. As shown in Table 6, **w/o Regularization** achieves lower AUC and AP than DPHIA without defense, indicating stronger attacks. However, its attack effectiveness decreases substantially under both defenses, whereas DPHIA consistently maintains stronger attacks. For example, on Citeseer, MD-filtering increases the AUC of **w/o Regularization** from 62.13% to 76.52%, while the AUC of DPHIA remains nearly unchanged. Although **w/o Regularization** achieves stronger attacks without defense, it significantly compromises stealthiness, highlighting the importance of adversarial training for attack stealth.

F.4 Parameter Sensitivity Analysis

To examine how dual-level adversarial regularization affects the trade-off between attack effectiveness and stealthiness, we vary $\alpha, \beta \in \{0, 0.25, 0.5, 0.75, 1\}$ and report HyGEN's AUC on Citeseer without defense and under MD-filtering and HD-filtering in Figure 6. Without defense, DPHIA maintains attack effectiveness across a broad range of parameter configurations. Under MD-filtering, increasing α with $\beta = 0$ improves attack effectiveness after filtering. Under HD-filtering, a moderate β with $\alpha = 0$ yields stronger attacks, whereas further increasing β weakens performance. These analyses provide practical guidance for parameter selection: a larger α is preferable against MD-filtering and a moderate β against HD-filtering. Overall, DPHIA remains effective across a broad parameter range, providing a favorable trade-off between attack effectiveness and stealthiness.

F.5 Efficiency Analysis

To assess efficiency, we compare the training time of DPHIA with that of HyperNear, the second-best baseline, across five datasets. Table 7 reports results from a single run. DPHIA is slightly slower on smaller datasets such as Citeseer and Cora, but faster on the larger DBLP and DBLP-A datasets. These results indicate that its adaptive ranking objective and dual-level regularization introduce only



(a) Without defense. (b) Under MD-filtering. (c) Under HD-filtering.

Figure 6: Sensitivity analysis of α and β (AUC) on Citeseer.

Table 7: Running time comparison (s).

Method	Citeseer	Cora	Pubmed	DBLP	DBLP-A
HyperNear	**12.49**	**12.04**	**42.57**	44.69	81.11
DPHIA	13.92	12.61	46.10	**41.36**	**78.83**

modest computational overhead. Overall, DPHIA remains computationally competitive, achieving efficiency comparable to that of HyperNear while providing improved attack effectiveness.

G Additional Related Works

G.1 Hyperedge Prediction

Hyperedge prediction aims to identify missing or future higher-order relations from an observed hypergraph and is commonly formulated as a classification problem over candidate node sets. Early methods projected hypergraphs into ordinary graphs and applied structural measures or conventional classifiers. For instance, Expansion (Yoon et al., 2020) constructs multiple n-projected graphs to progressively incorporate higher-order information. Set-based methods, including DeepSets (Zaheer et al., 2017), HyperSAGNN (Zhang et al., 2019), and NHP (Yadati et al., 2020), treat each candidate as an unordered node set and learn a permutation-invariant representation for classification. However, by encoding candidates independently, they cannot explicitly capture dependencies across hyperedges. Hypergraph Neural Networks (HNNs) address this limitation by performing message passing over the node–hyperedge incidence structure (Feng et al., 2019; Dong et al., 2020; Chien et al., 2021), thereby modeling both intra-hyperedge and inter-hyperedge interactions. These advances have led to hypergraph-based predictors that use HNNs to derive structure-aware node representations and aggregate them to classify candidate sets. AHP (Hwang et al., 2022) combines an HNN-based predictor with adversarial training to replace heuristic negative sampling with learnable negative generation. CASH (Ko et al., 2025) integrates context-aware aggregation with self-supervised learning to improve candidate representations under sparse observations. HyGEN (Yu et al., 2025) further enhances negative hyperedge generation through positive guidance and false-negative regularization. More recently, HyperNoRA (Li et al., 2026c) improves prediction through global node relation modeling and contrastive learning. S^3Hyper (Li et al., 2026b) addresses sparse supervision and overlooked hyperedge substructures through substructure-aware aggregation and adaptive contrastive learning. Despite improvements in prediction accuracy, the robustness of hyperedge prediction remains largely underexplored (Li et al., 2026a).