Noise Robust Graph Learning under Feature-Dependent Graph-Noise

Yeonjun In
yeonjun.in@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Kanghoon Yoon
ykhoon08@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Sukwon Yun
swyun@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Kibum Kim
kb.kim@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

Sungchul Kim
sukim@adobe.com
Adobe Research
San Jose, CA, USA

Chanyoung Park*
cy.park@kaist.ac.kr
KAIST
Daejeon, Republic of Korea

ABSTRACT

In practical situations, node features often contain noise from various sources, leading to severe performance degradation of GNNs. While several methods aim to improve robustness, they often make an unrealistic assumption that the noise in node features is independent of the graph structure and node labels, limiting their applicability in real-world scenarios. To this end, we newly present a more realistic noise scenario called feature-dependent graph-noise (FDGN), where noisy node features may entail both structure and label noise. Furthermore, we propose a deep generative model that directly captures the causal relationships among variables in the DGP of FDGN. We establish a tractable and feasible learning objective based on variational inference and thoroughly discuss the instantiations of model components corresponding to the derived learning objective. Our proposed method, PRINGLE, outperforms baselines on widely used benchmark datasets and our newly introduced real-world graph datasets that simulate FDGN in e-commerce systems.

CCS CONCEPTS

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

KEYWORDS

graph noise, noise robust graph learning, generative graph learning

ACM Reference Format:
Yeonjun In, Kanghoon Yoon, Sukwon Yun, Kibum Kim, Sungchul Kim, and Chanyoung Park. 2024. Noise Robust Graph Learning under Feature-Dependent Graph-Noise. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3589335.3651916

*Corresponding author.

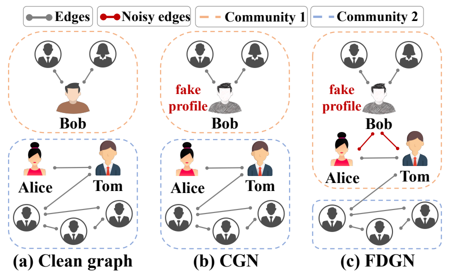
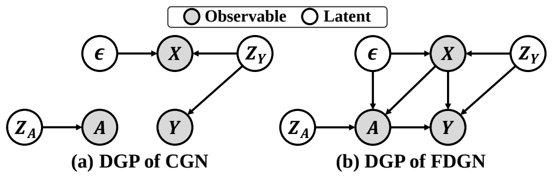


Figure 1: Social network examples.



$$\begin{tikzcd} \epsilon \arrow[r] & X & Z_Y \arrow[l] \arrow[dl] \\ Z_A \arrow[r] & A & Y \end{tikzcd}$$

$$\begin{tikzcd} & \epsilon \arrow[r] \arrow[d] & X \arrow[dl] \arrow[d] & Z_Y \arrow[l] \arrow[dl] \\ Z_A \arrow[r] & A \arrow[r] & Y & \end{tikzcd}$$

Figure 2: A directed graphical model indicating a DGP of (a) CGN, and (b) FDGN.

1 INTRODUCTION

In the majority of real-world scenarios, node features frequently exhibit noise due to various factors, leading to the creation of inaccurate graph representations [12, 19]. For instance, in social networks, users may create fake profiles or posting, resulting in noisy node features. Similarly, in product co-purchase networks found on e-commerce systems, node features can be contaminated by fake reviews. Recent studies have revealed the vulnerability of GNNs to such scenarios, highlighting the necessity to design robust GNN models against noisy node features. To this end, various methods have been proposed to make a huge success in terms of model robustness [12, 19].

While such existing robust GNN models have proven effective, we argue that their practicality is restricted by an unrealistic assumption regarding graph noise: *they assume that the noise in node features is independent of the graph structure or node labels.* For example, in the conventional graph noise (CGN) assumption in terms of node features (Fig. 1(b)), Bob's fake profile does not influence other nodes, which is also explained by the data generating process (DGP) of CGN (See Fig. 2(a)) in which no causal relationships exist among the noisy node features X, graph structure A, and node labels Y.

However, in reality (See Fig. 1(c)), other users may make connections with Bob based on his fake profile (i.e., structure noise), which may also eventually change their community (i.e., label noise), and such causal relationships among X, A, and Y (i.e., $A \leftarrow X$, $Y \leftarrow X$, and $Y \leftarrow A$) are depicted in Fig. 2(b).

This noise scenario is commonly encountered in various real-world applications. Consider another real-world scenario, such as a product co-purchase network on an e-commerce platform, where nodes represent products and edges represent the co-purchase relationship between products. In this case, fake reviews on products written by a fraudster would make other users purchase irrelevant products, which adds irrelevant edges between products (i.e., structure noise). Consequently, this would make the automated product category labeling system to inaccurately annotate product categories (i.e., label noise), as it relies on the node features and the graph structure, both of which are contaminated.

These examples demonstrate that, in reality, ***noisy node features may entail both structure and label noise***. However, we observe that existing robust GNN models indeed fail to generalize effectively in such a noise scenario due to their inadequate assumption on the data generation process (Sec 3.1).

In this work, to reflect such a realistic scenario in graph learning, we newly introduce **f**eature-**d**ependent **g**raph-**n**oise (FDGN) and propose a **pri**ncipled **n**oisy **g**raph **le**arning framework (PRINGLE), which models the DGP of a more realistic FDGN assumption. We first illustrate the DGP of FDGN as shown in Fig. 2(b). More precisely, we introduce three observable variables (i.e., node features X, graph structure A, and node labels Y) and three latent variables (i.e., noise incurring variable ϵ, latent clean graph structure Z_A, and latent clean node labels Z_Y), while defining causal relationships among these variables to represent the data generation process of FDGN. We then devise a deep generative model that directly captures the causal relationships among the variables in the DGP of FDGN, and derive a tractable and feasible learning objective based on variational inference. It is worth noting that although the main focus of this paper is on the node feature noise and its influence across the graph, our proposed robust graph learning framework is capable of generalizing not only to feature-dependent graph noise (FDGN), but also to independent structure/feature/label noise that is also prevalent in real-world applications. This implies that PRINGLE has a wider range of applicability than existing robust GNN models. Moreover, to conduct a rigorous evaluation of PRINGLE, we conduct evaluations not only on existing benchmark datasets, but also on our newly introduced real-world graph datasets that simulate FDGN within e-commerce systems, providing a valuable alternative to synthetic settings.

In summary, the main contributions of our paper are three-fold:

- We investigate limitations of the conventional graph noise assumption in terms of node features, and introduce a more realistic graph noise scenario, **f**eature-**d**ependent **g**raph-**n**oise (FDGN). To the best of our knowledge, this is the first attempt to understand the data generation process in graph domain that mimics the noise scenario in real-world.
- We propose the **pri**ncipled **n**oisy **g**raph **le**arning framework (PRINGLE) addressing FDGN by modeling its DGP. PRINGLE outperforms state-of-the-art baselines in node classification and link prediction tasks under various scenarios including feature-dependent graph-noise and independent structure/feature/label noise.
- In addition to existing benchmark datasets in which noise is synthetically generated, we further introduce novel graph benchmark datasets that simulate FDGN within e-commerce systems, which is expected to foster practical research in noise-robust graph learning.

2 RELATED WORK

The objective of noise-robust graph learning is to train GNN models when the input graph data exhibits one or more of the following types of noise: 1) node feature noise, 2) graph structure noise, and/or 3) node label noise. The majority of existing approaches focus primarily on graphs containing only a single type of noise.

Feature noise-robust graph learning. To address the noisy node features, various approaches including adversarial training [26] and test-time graph transformation [12], have been proposed. Additionally, recent studies have highlighted the significance of fully leveraging structural information. AirGNN [19] proposed a novel message passing mechanism that identifies the nodes with noisy features and learns node-wise adaptive coefficients that balance the feature aggregation and use of their own noisy features. The identification process is guided by the intuition that nodes with noisy features tend to have dissimilar features within their local neighborhoods. In summary, this approach tackles the noisy node features while assuming that the structure of the input graph is noise-free.

Structure noise-robust graph learning. To address the noisy graph structure, various approaches including robust message passing scheme [15] and graph structure learning [11] have been proposed. Among them, a representative approach is based on the graph structure learning (GSL), which aims to learn a refined graph structure from a given graph. Specifically, RSGNN [6] aims to train a graph structure learner, which is composed of an MLP encoder and a regularizer for enhancing feature-smoothness, which encourages the nodes with similar features to be connected in the refined structure. STABLE [17] aims at acquiring robust node representations by roughly refining the given graph by removing easily detectable noisy edges, typically those connecting nodes with low feature-similarity, after which node representations are learned in an unsupervised manner. Finally, based on these representations, a kNN graph is constructed to serve as the refined structure. In summary, these methods tackle the noisy graph structure while assuming that node features are noise-free.

Label noise-robust graph learning. There are numerous studies that address the noisy labels on non-graph data [16, 27]. However, since they are not directly applicable for graph data [5], recent studies investigated the label noise-robust graph learning methods. The key idea of NRGNN [5] is to correct the predictions of unlabeled nodes affected by information propagation from falsely labeled nodes. To do so, NRGNN learns a new graph structure where two nodes with similar features are connected, based on the assumption that two nodes are more likely to have the same label if they have similar features. This strategy mitigates the information propagation from falsely labeled nodes. RTGNN [24] identifies clean labeled samples from noisy labeled ones based on the small-loss criteria and leverages pseudo-labeling to supplement the labeled nodes. However, nodes with noisy features or structures would yield a large-loss even if their labels do not contain any noise, which results in inaccuracies in the selection of clean labeled samples. Therefore, these methods tackle the noisy node labels while assuming that both node features and graph structure are noise-free.

Generative graph learning. Apart from the noise-robust graph learning methods, several studies adopted deep generative modeling to infer latent graph data [7, 14, 20]. Most recently, WSGNN [14] uses a probabilistic generative approach and variational inference to infer the latent graph structure and node labels. GraphGLOW [28] aims at inferring the latent graph structure that can be transferred to the out-of-domain by utilizing a probabilistic generative approach and variational inference. However, they assume noise-free graphs, making it less effective in handling various noise scenarios commonly found in real-world applications.

In summary, each of the aforementioned methods assumes the completeness of at least one of the data sources, i.e., node features, graph structures, or node labels. In contrast, our proposed method is constructed under a more realistic FDGN assumption, where noise in node features may result in structural and label noise. This fundamental difference liberates the proposed method from the limited assumptions of existing methods.

3 FEATURE-DEPENDENT GRAPH-NOISE

Definition. In this section, we define a realistic graph noise assumption, i.e., FDGN, and its DGP. Specifically, we assume that the graph containing FDGN is generated according to the graphical model in Fig. 2(b). We first introduce each variable in the graphical model, and then explain each relationship between variables through two real-world applications where FDGN is prevalent: social networks (i.e., user-user graph) and co-purchase networks (i.e., product-product graph) within e-commerce systems.

In the graphical model in Fig. 2(b), X represents the node features, which may contain noisy node features; Y represents the observed node labels, which may contain noisy labels; A represents the observed edges between two nodes, which may contain noisy edges; ϵ represents the environment variable that causes the noise; Z_Y represents the latent clean node labels; Z_A represents the latent clean graph structure that contains all potential connections between nodes. We provide explanations for each relationship in the graphical model of FDGN shown in Fig. 2(b):

- $\underline{X \leftarrow (\epsilon, Z_Y)}$: ϵ and Z_Y are causes of X. In social networks, users create their profiles and postings (i.e., X) regarding their true communities or interests (i.e., Z_Y). However, if users decide to display fake profiles for some reason (i.e., ϵ), ϵ is a cause of the noisy node features X. In co-purchase networks, the reviews and descriptions of products (i.e., X) are written regarding their true categories (i.e., Z_Y). However, if a fraudster (i.e., ϵ) writes fake reviews on products, ϵ is a cause of the noisy node features (i.e., X).
- $\underline{A \leftarrow (Z_A, X)}$: Z_A and X are causes of A. In social networks, the follow relationship among users (i.e., A) are made based on their latent relationships (i.e., Z_A). However, if a user creates a fake profile (i.e., X), some irrelevant users may follow the user based on his/her fake profile, which leads to noisy edges (i.e., A). In co-purchase networks, the co-purchase relationship among products (i.e., A) are made based on their true relevance (i.e., Z_A). However, fake reviews on products written by a fraudster (i.e., X) would make other users purchase irrelevant products, which leads to noisy edges (i.e., A).
- $\underline{A \leftarrow \epsilon}$: To provide a broader scope, we also posit that ϵ is a potential cause of A. This extension is well-founded, as real-world applications often exhibit graph structure noise originating from various sources in addition to the feature-dependent noise [8, 18].
- $\underline{Y \leftarrow (Z_Y, X, A)}$: Z_Y, X, and A are causes of Y. In social networks, the true communities (or interests) of users (i.e., Z_Y) are leveraged to promote products to targeted users within a community [21]. To detect the communities, both node features and graph structures are utilized. However, if a user has noisy node features (i.e., X) or noisy edges (i.e., A), the user may be assigned to a wrong community (or interest), which leads to noisy labels (i.e., Y). In co-purchase networks, machine learning-based automated labeling techniques are widely used in e-commerce systems to label the true categories of products (i.e., Z_Y) since new products are continuously released. However, the automated labeling systems may become inaccurate due to noisy node features (i.e., X) and noisy graph structures (i.e., A), which leads to noisy node labels (i.e., Y).

For simplicity, we assume that ϵ is not a cause of Y. This assumption aligns with practical scenarios in real-world applications, where an instance is more likely to be mislabeled due to confusing or noisy features rather than arbitrary sources, i.e., instance-dependent label-noise [2, 27]. In other words, label noise in graphs is predominantly caused by confusing or noisy features and graph structure (i.e., $Y \leftarrow (X, A)$), rather than an arbitrary external factor (i.e., $Y \nleftarrow \epsilon$).

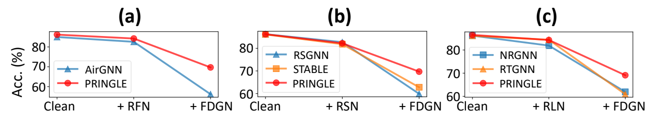


Figure 3: Node classification performance of baselines and PRINGLE on Cora dataset. Here, *+ RFN*, *+ RSN*, and *+ RLN* indicate injecting the random feature noise, random structure noise, and random label noise into the original graph, respectively.

3.1 Preliminary Analysis on FDGN

We conduct an analysis to examine how well existing robust GNN models generalize to FDGN. We generate three types of noise: random feature noise [19], random structure noise [17], and random label noise [5, 24], following the convention of the existing studies. As baselines, we consider (a) AirGNN as a feature noise-robust graph learning method, (b) RSGNN and STABLE as structure noise-robust graph learning methods, and (c) NRGNN and RTGNN as label noise-robust graph learning methods.

We observe that while existing noise-robust graph learning methods perform well under the random feature noise (i.e., + RFN in Fig. 3(a)), structure noise (+ RSN in Fig. 3(b)), and label noise (+ RLN in Fig. 3(c)), their performance significantly drops under FDGN (i.e., + FDGN in Fig. 3). In contrast, our proposed method (PRINGLE) demonstrates competitive results under the random noise scenarios, while notably outperforming the baselines under FDGN. This points to a key distinction – existing noise-robust graph learning methods struggle to generalize to FDGN due to their limited assumptions regarding the graph noise. Specifically, as summarized in Sec. 2, each category of methods assumes at least one of the data sources is noise-free: node features, graph structures, or node labels. Nevertheless, the causal relationships among X, A, and Y within the data generation process of FDGN gives rise to scenarios involving concurrent feature, structure, and label noise. Consequently, existing robust GNN models fall short of effectively generalizing to

FDGN, as they overlook such underlying relationships among noise types, leading to model designs assuming the completeness of at least one data source. Conversely, PRINGLE directly captures the underlying relationships by modeling the DGP of FDGN, resulting in superior generalization to FDGN.

4 PROPOSED METHOD: PRINGLE

In this section, we propose a principled noisy graph learning framework (PRINGLE) to tackle a more realistic noise scenario, FDGN. It is essential to highlight that under FDGN, noisy node features entail both structure and label noise, resulting in a graph that does not contain any noise-free data sources, i.e., a graph with noisy X, noisy A, and noisy Y. This point presents a non-trivial challenge for the existing noise-robust graph learning methods to tackle FDGN, as they assume the completeness of at least one data source. To address this challenge, we design a deep generative model that directly models the DGP of FDGN, thereby capturing the causal relationships among the variables that introduce noise. First, we derive the Evidence Lower Bound (ELBO) for the observed data log-likelihood $P(X, A, Y)$ based on the graphical model of FDGN (**Section 4.2**). Subsequently, we discuss model instantiations for the model components corresponding to the derived terms, including implementation details (**Section 4.3**). It is essential to highlight that our approach can handle both node classification and link prediction tasks, making it versatile and applicable in various situations.

4.1 Problem Statement

Notations We have an undirected and unweighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ where $\mathcal{V} = \{v_1, ..., v_N\}$ represents the set of nodes and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ indicates the set of edges. Each node v_i has the node features $\mathbf{X}_i \in \mathbb{R}^F$ and node labels $\mathbf{Y}_i \in \{0,1\}^C$, where F is the number of features for each node and C indicates the number of classes. We represent the observed graph structure using the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where $\mathbf{A}_{ij} = 1$ if there is an edge connecting nodes v_i and v_j, and $\mathbf{A}_{ij} = 0$ otherwise.

Tasks: node classification and link prediction In the node classification task, we assume the semi-supervised setting where only a portion of nodes are labeled (i.e., $\mathcal{V}^L$). Our objective is to predict the labels of unlabeled nodes (i.e., $\mathcal{V}^U$) by inferring the latent clean node label Z_Y. In the link prediction task, our goal is to predict reliable links based on partially observed edges by inferring the latent clean graph structure Z_A. It is important to note that, according to the FDGN assumption, the observed node features, graph structure, and node labels may contain noise.

4.2 Model Formulation

We commence by modeling joint distribution $P(X, A, Y)$. We assume that the joint distribution $P(X, A, Y)$ is differentiable nearly everywhere regarding both θ and the latent variables (ϵ, Z_A, Z_Y). Note that the generative parameter θ serves as the decoder network that models the distribution $P(X, A, Y)$. The joint distribution of $P(X, A, Y)$ can be represented as:

$$p_\theta(X, A, Y) = \int_\epsilon \int_{Z_A} \int_{Z_Y} p_\theta(X, A, Y, \epsilon, Z_A, Z_Y) d\epsilon dZ_A dZ_Y. \quad (1)$$

However, computing this evidence integral is either intractable to calculate in closed form or requires exponential time. As the evidence integral is intractable for computation, calculating the conditional distribution of latent variables $p_\theta(\epsilon, Z_A, Z_Y | X, A, Y)$ is also intractable: $p_\theta(\epsilon, Z_A, Z_Y | X, A, Y) = \frac{p_\theta(X,A,Y,\epsilon,Z_A,Z_Y)}{p_\theta(X,A,Y)}$.

To infer the latent variables, we introduce an inference network ϕ to model the variational distribution $q_\phi(\epsilon, Z_A, Z_Y | X, A, Y)$, which serves as an approximation to the posterior $p_\theta(\epsilon, Z_A, Z_Y | X, A, Y)$. To put it more concretely, the posterior distribution can be decomposed into three distributions determined by trainable parameters ϕ_1, ϕ_2, and ϕ_3. Based on the observed conditional independence relationships [1], we decompose $q_\phi(\epsilon, Z_A, Z_Y | X, A, Y)$ as follows:

$$q_\phi(\epsilon, Z_A, Z_Y | X, A, Y) = q_{\phi_1}(Z_A | X, A, \epsilon) q_{\phi_2}(\epsilon | X, A, Z_Y) q_{\phi_3}(Z_Y | X, A, Y). \quad (2)$$

For simplicity, we introduce two additional assumptions. First, when the node features X and observed graph structure A are given, latent clean graph structure Z_A is conditionally independent from the noise-incurring variable ϵ, i.e., $q_{\phi_1}(Z_A | X, A, \epsilon) = q_{\phi_1}(Z_A | X, A)$. Second, when X and A are given, latent clean labels Z_Y is conditionally independent from the observed node labels Y, i.e., $q_{\phi_3}(Z_Y | X, A, Y) = q_{\phi_3}(Z_Y | X, A)$. This approximation, known as the mean-field method, is a prevalent technique utilized in variational inference-based methods [14, 20]. As a result, we can simplify Eqn. 2 as follows:

$$q_\phi(\epsilon, Z_A, Z_Y | X, A, Y) = q_{\phi_1}(Z_A | X, A) q_{\phi_2}(\epsilon | X, A, Z_Y) q_{\phi_3}(Z_Y | X, A). \quad (3)$$

To jointly optimize the parameter ϕ and θ, we adopt the variational inference framework [3, 27] to optimize the Evidence LowerBOund (ELBO) of the marginal likelihood for observed data, rather than optimizing the marginal likelihood directly. Specifically, we derive the negative ELBO, i.e., $\mathcal{L}_{\text{ELBO}}$, as follows:

$$\begin{aligned}
\mathcal{L}_{\text{ELBO}} &= -\mathbb{E}_{Z_A \sim q_{\phi_1}(Z_A|X,A)} \mathbb{E}_{\epsilon \sim q_{\phi_2}(\epsilon|X,A,Z_Y)} \left[\log(p_{\theta_1}(A | X, \epsilon, Z_A))\right] \\
&- \mathbb{E}_{\epsilon \sim q_{\phi_2}(\epsilon|X,A,Z_Y)} \mathbb{E}_{Z_Y \sim q_{\phi_3}(Z_Y|X,A)} \left[\log(p_{\theta_2}(X | \epsilon, Z_Y))\right] \\
&- \mathbb{E}_{Z_Y \sim q_{\phi_3}(Z_Y|X,A)} \left[\log(p_{\theta_3}(Y | X, A, Z_Y))\right] \\
&+ kl(q_{\phi_3}(Z_Y | X, A) || p(Z_Y)) + kl(q_{\phi_1}(Z_A | X, A) || p(Z_A)) \\
&+ \mathbb{E}_{Z_Y \sim q_{\phi_3}(Z_Y|X,A)} \left[kl(q_{\phi_2}(\epsilon | X, A, Z_Y) || p(\epsilon))\right]. \quad (4)
\end{aligned}$$

where $kl(\cdot||\cdot)$ denotes KL divergence. The encoders ϕ_1, ϕ_2, and ϕ_3 infer three latent variables, while θ_1, θ_2, and θ_3 are decoder networks used for generating three observable variables. Our objective is to find the optimal values of $\phi = \{\phi_1, \phi_2, \phi_3\}$ and $\theta = \{\theta_1, \theta_2, \theta_3\}$ that minimize the value of $\mathcal{L}_{\text{ELBO}}$, expressed as $\text{argmin}_{\theta,\phi} \mathcal{L}_{\text{ELBO}}$. By doing so, the encoders and decoders are trained to directly capture the causal relationships among the variables that introduce noise. Consequently, it promotes the accurate inference of the latent clean node label Z_Y and latent clean graph structure Z_A to effectively perform the node classification and link prediction tasks even in the presence of FDGN.

4.3 Model Instantiations

In this section, we present the details of the practical implementation and optimization of PRINGLE based on the learning objective, $\mathcal{L}_{\text{ELBO}}$. The overall architecture of PRINGLE are provided in Fig 4. The key challenge of the instantiation is how to accurately infer the latent variables Z_A, Z_Y, and ϵ in the presence of noisy X, A, and Y. To alleviate the challenge, we design the robust inference encoders (Fig 4(a)) and generative decoders (Fig 4(b)) with the corresponding

[1]We observe the following conditional independence relationships in Fig. 2(b): (1) $Z_A \perp Y | X, A, \epsilon$, (2) $Z_A \perp Z_Y | A, X, \epsilon$, (3) $\epsilon \perp Y | Z_Y, X, A$.

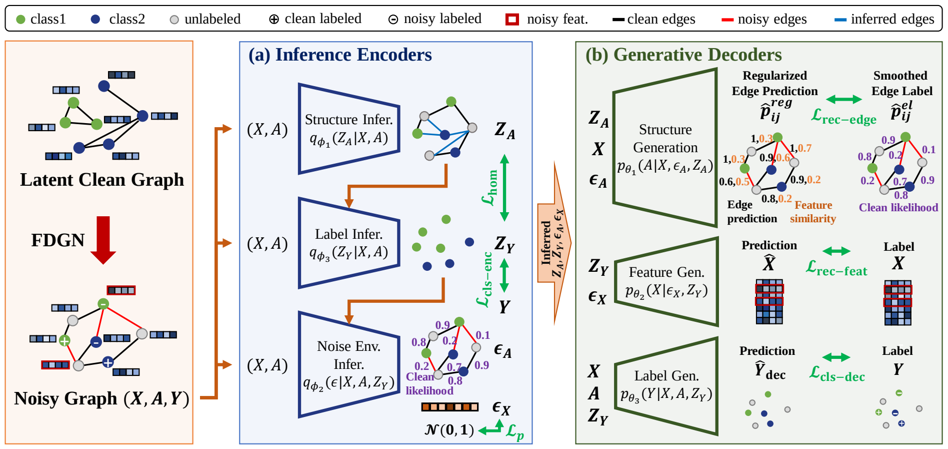

class1
class2
unlabeled
clean labeled
noisy labeled
noisy feat.
clean edges
noisy edges
inferred edges
Latent Clean Graph
FDGN
Noisy Graph (X, A, Y)
(a) Inference Encoders
(X, A)
Structure Infer. $q_{\phi_1}(Z_A|X,A)$
Z_A
$\mathcal{L}_{\text{hom}}$
Label Infer. $q_{\phi_3}(Z_Y|X,A)$
Z_Y
$\mathcal{L}_{\text{cls-enc}}$
Y
Noise Env. Infer. $q_{\phi_2}(\epsilon|X,A,Z_Y)$
Clean likelihood
ϵ_A
ϵ_X
$\mathcal{N}(0,1)$
$\mathcal{L}_p$
Inferred $Z_A, Z_Y, \epsilon_A, \epsilon_X$
(b) Generative Decoders
Z_A X ϵ_A
Structure Generation $p_{\theta_1}(A|X,\epsilon_A,Z_A)$
Regularized Edge Prediction $\hat{p}_{ij}^{reg}$
$\mathcal{L}_{\text{rec-edge}}$
Smoothed Edge Label $\hat{p}_{ij}^{el}$
Edge prediction
Feature similarity
Clean likelihood
Z_Y ϵ_X
Feature Gen. $p_{\theta_2}(X|\epsilon_X,Z_Y)$
Prediction $\hat{X}$
$\mathcal{L}_{\text{rec-feat}}$
Label X
X A Z_Y
Label Gen. $p_{\theta_3}(Y|X,A,Z_Y)$
Prediction $\hat{Y}_{\text{dec}}$
$\mathcal{L}_{\text{cls-dec}}$
Label Y

Figure 4: Overall architecture of PRINGLE. (a) With the noisy graph (X, A, Y) as inputs, we design the inference encoders (ϕ_1, ϕ_2 and ϕ_3) and regularizers ($\mathcal{L}_{\text{hom}}$, $\mathcal{L}_{\text{cls-enc}}$, and $\mathcal{L}_{\text{p}}$) to effectively infer the latent variables Z_A, Z_Y, ϵ_A, and ϵ_X. (b) Leveraging the inferred latent variables, we formulate the generative decoders (θ_1, θ_2, and θ_3) and reconstruction loss functions ($\mathcal{L}_{\text{rec-edge}}$, $\mathcal{L}_{\text{rec-feat}}$, and $\mathcal{L}_{\text{cls-dec}}$) to capture the causal relationships that generate noise in the graph.

regularizers (Fig 4(a)) and reconstruction losses (Fig 4(b)). Consequently, the encoders would be able to accurately infer the latent variables by capturing the causal relationships among the variables that introduce noise.

4.3.1 *Modeling Inference Encoder.*

In this section, we describe the implementations of the encoders, i.e., ϕ_1, ϕ_3, and ϕ_2, that aim to infer the latent variables, i.e., Z_A, Z_Y, and ϵ, respectively.

Modeling $q_{\phi_1}(Z_A|X, A)$. The objective of modeling $q_{\phi_1}(Z_A|X, A)$ is to accurately infer the latent clean graph structure Z_A that enhances the message passing of a GNN model and the performance of link prediction. To this end, we design the encoder ϕ_1 as a graph structure learner to accurately infer the latent graph structure Z_A. More specifically, we use a GCN encoder to acquire deterministic node embeddings, i.e., $\mathbf{Z} = \text{GCN}_{\phi_1}(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{N \times d_1}$, where d_1 is the dimension of node embedding. To obtain the latent graph $\hat{\mathbf{A}} = \{\hat{a}_{ij}\}_{N\times N}$, we sample from $\hat{a}_{ij} \sim \text{Bern}(\hat{p}_{ij})$ where the estimated parameter $\hat{p}_{ij}$ is computed as follows: $\hat{p}_{ij} = \rho(s(\mathbf{Z}_i, \mathbf{Z}_j))$, where $s(\cdot,\cdot)$ is a cosine similarity function and ρ is the ReLU activation function.

To further promote an accurate inference of Z_A, we inject our prior knowledge, i.e., $p(Z_A)$, into the inference procedure: the latent graph structure predominantly consists of assortative edges that have the potential to enhance feature propagation within GNNs [28]. In recent studies [4, 6], the γ-hop subgraph similarity has served as a potent metric for identifying assortative edges. Hence, we aim to regularize Z_A to align with our prior knowledge by minimizing the KL divergence between the latent graph structure $\hat{\mathbf{A}}$ and prior graph structure $\mathbf{A}^{\text{p}} = \{a^{\text{p}}_{ij}\}_{N\times N}$, where a^{p}_{ij} is sampled from a Bernoulli distribution with a probability as given by the γ-hop subgraph similarity. Such regularization leads the estimated probability $\hat{p}_{ij}$ between two nodes to increase if they exhibit a high subgraph similarity. It is important to note that this regularization is equivalent to minimizing $kl(q_{\phi_1}(Z_A|X, A)||p(Z_A))$ in Eqn. 4.

However, computing $\hat{p}_{ij}$ in every epoch is impractical for large graphs, i.e., $O(N^2)$. To mitigate the issue, we pre-define a candidate graph that consists of the observed edge set $\mathcal{E}$ and a k-NN graph based on the γ-hop subgraph similarity. We denote the set of edges in the k-NN graphs as $\mathcal{E}^{\gamma}_{k}$. Then, we compute the $\hat{p}_{ij}$ values of the edges in a candidate graph, i.e., $\mathcal{E}^{\gamma}_{k} \cup \mathcal{E}$, instead of all edges in $\{(i, j)|i \in \mathcal{V}, j \in \mathcal{V}\}$, to estimate the latent graph structure denoted as $\hat{\mathbf{A}}$. It is important to highlight that obtaining $\mathcal{E}^{\gamma}_{k}$ is carried out offline before model training, thus incurring no additional computational overhead during training. This implementation technique achieves a similar effect as minimizing $kl(q_{\phi_1}(Z_A|X, A)||p(Z_A))$ while significantly addressing computational complexity from $O(N^2)$ to $O(|\mathcal{E}^{\gamma}_{k} \cup \mathcal{E}|)$, where $N^2 \gg |\mathcal{E}^{\gamma}_{k} \cup \mathcal{E}|$.

Modeling $q_{\phi_3}(Z_Y|X, A)$. The objective of modeling $q_{\phi_3}(Z_Y|X, A)$ is to accurately infer the latent clean node label Z_Y that enhances the performance of node classification with the help of the inferred Z_A. To this end, we instantiate the encoder ϕ_3 as a GCN classifier that deterministically outputs the probability of the latent label Z_Y. Specifically, we infer Z_Y through $\hat{\mathbf{Y}} = \text{GCN}_{\phi_3}(\mathbf{X}, \hat{\mathbf{A}}) \in \mathbb{R}^{N\times C}$ because $\hat{\mathbf{A}}$ contains not only $\mathbf{A}$, but also rich structural information that is missing in $\mathbf{A}$, which helps enhance the inference of Z_Y. We introduce the node label classification loss $\mathcal{L}_{\text{cls-enc}} = \sum_{i\in\mathcal{V}^L} \text{CE}(\hat{\mathbf{Y}}_i, \mathbf{Y}_i)$, where CE is the cross entropy loss.

To further enhance the quality of inference of Z_Y, we inject our prior knowledge, i.e., $p(Z_Y)$, into the inference procedure: two nodes connected on the latent graph structure Z_A are expected to have an identical latent labels Z_Y, known as class homophily [23]. Hence, we aim to regularize Z_Y to align with our prior knowledge by minimizing the KL divergence between the probability predictions $\hat{\mathbf{Y}}$ of each node and its first order neighbors in the estimated latent structure $\hat{\mathbf{A}}$. The implemented loss function is given by: $\mathcal{L}_{\text{hom}} = \sum_{i\in\mathcal{V}} \frac{\sum_{j\in N_i} \hat{p}_{ij}\cdot kl(\hat{\mathbf{Y}}_j||\hat{\mathbf{Y}}_i)}{\sum_{j\in N_i} \hat{p}_{ij}}$, where N_i denotes the set of first-order neighbors of node v_i within the estimated latent structure $\hat{\mathbf{A}}$. It is worth noting that this regularization is equivalent to minimizing $kl(q_{\phi_3}(Z_Y|X, A)||p(Z_Y))$ in Eqn. 4.

Modeling $q_{\phi_2}(\epsilon|X, A, Z_Y)$. To model $q_{\phi_2}(\epsilon|X, A, Z_Y)$, we simplify $q_{\phi_2}(\epsilon|X, A, Z_Y)$ into $q_{\phi_{21}}(\epsilon_X|X, Z_Y)$ and $q_{\phi_{22}}(\epsilon_A|X, A)$, where ϵ_X

and ϵ_A are independent variables that incur the feature and structure noise, respectively.

The objective of modeling $q_{\phi_{22}}(\epsilon_A|X, A)$ is to accurately infer the structure noise incurring variable ϵ_A that determines whether each edge is clean or noisy. To this end, we regard ϵ_A as a set of scores indicating the likelihood of each observed edge being clean or noisy. To estimate the likelihood, we draw inspiration from an early-learning phenomenon [1]. This phenomenon indicates that deep neural networks have a tendency to initially focus on learning from training data with clean labels during an "early learning" phase. In other words, during the early learning phase, deep neural networks are trained to yield small loss values on the clean labeled data. Building upon this well-established observation, we compute the set of link prediction losses using MSE on the observed edges $\mathcal{E}$ as $\{(1-\hat{p}_{ij}^{el})^2|(i,j)\in\mathcal{E}\}$, where $\hat{p}_{ij}^{el}$ represents the $\hat{p}_{ij}$ value at the final epoch during early-learning phase. Therefore, an edge with high $\hat{p}_{ij}^{el}$ value can be considered as a clean edge, and we instantiate ϵ_A as $\{\hat{p}_{ij}^{el}|(i,j)\in\mathcal{E}\}$. However, the loss-based criteria $\hat{p}_{ij}^{el}$ may introduce uncertainty in identifying clean edges as it relies on a single training point's value. In other words, $q_{\phi_{22}}(\epsilon_A|X, A)$ follows an unknown distribution with high variance. To reduce uncertainty of the inferred ϵ_A, we regularize $q_{\phi_{22}}(\epsilon_A|X, A)$ to follow the same distribution with low variance, i.e., $p(\epsilon_A)$, which is equivalent to reducing $kl(q_{\phi_{22}}(\epsilon_A|X,A)||p(\epsilon_A))$. To implement it, we adopt an exponential moving average (EMA) technique: $\hat{p}_{ij}^{el} \leftarrow \xi\hat{p}_{ij}^{el} + (1-\xi)\hat{p}_{ij}^{c}$, where $\hat{p}_{ij}^{c}$ indicates the value of $\hat{p}_{ij}$ at the current training point, and ξ indicates the decaying hyperparameter fixed to 0.9. Please note that we scale $\hat{p}_{ij}^{el}$ with a minimum value of 0.9 and a maximum value of 1.

For the encoder ϕ_{21}, we use an MLP that takes $CONCAT(X, Z_Y)$ as an input and outputs ϵ_X. Additionally, we regularize $p(\epsilon_X)$ to follow the standard multivariate normal distribution, which means that a closed form solution of $kl(q_{\phi_{21}}(\epsilon_X|X, Z_Y)||p(\epsilon_X))$ can be obtained as $\mathcal{L}_{\text{p}} = -\frac{1}{2}\sum_{j=1}^{d_2}(1+\log\sigma_j^2-\mu_j^2-\sigma_j^2)$, where d_2 is the dimension of a ϵ_X [13]. Note that these two regularization techniques are equivalent to minimizing $\mathbb{E}_{Z_Y\sim q_{\phi_3}}\left[kl(q_{\phi_2}(\epsilon|X,A,Z_Y)||p(\epsilon))\right]$ in Eqn. 4.

4.3.2 *Modeling Generative Decoder.*

In this section, we describe the implementations of the decoders, i.e., θ_1, θ_2, and θ_3, that generate the observable variables, i.e., A, X, and Y, respectively.

Modeling $p_{\theta_1}(A|X,\epsilon,Z_A)$. The probability $p(A|X,\epsilon,Z_A)$ means the likelihood of how well the noisy edge A is reconstructed from the latent graph structure Z_A along with ϵ and X. Hence, we design the decoder θ_1 as an edge reconstruction model. We aim to maximize $\log(p_{\theta_1}(A|X,\epsilon,Z_A))$ (or equivalently minimize the negative log likelihood) to discover the latent graph structure Z_A from which the noisy edge A is reconstructed given noise sources, i.e., X and ϵ. It is important to note that such a learning objective is equivalent to minimizing $-\mathbb{E}_{Z_A\sim q_{\phi_1}}\mathbb{E}_{\epsilon\sim q_{\phi_2}}\left[\log(p_{\theta_1}(A|X,\epsilon,Z_A))\right]$ in Eqn. 4, which is implemented as an edge reconstruction loss forcing the estimated latent structure $\hat{\mathbf{A}}$ to assign greater weights to clean edges and reduce the influence of noisy edges, which is defined as $\mathcal{L}_{\text{rec-edge}}$:

$$\mathcal{L}_{\text{rec-edge}} = \frac{N}{|\mathcal{E}|+|\mathcal{E}^-|}\left(\sum_{(i,j)\in\mathcal{E}}(\hat{p}_{ij}^{reg}-\hat{p}_{ij}^{el})^2 + \sum_{(i,j)\in\mathcal{E}^-}(\hat{p}_{ij}-0)^2\right), \tag{5}$$

where $\mathcal{E}$ and $\mathcal{E}^-$ denote the positive edges and randomly sampled negative edges, respectively. To compute $\mathcal{L}_{\text{rec-edge}}$, we employ regularizations on both the predictions (i.e., $\hat{p}_{ij}^{reg}$) and labels (i.e., $\hat{p}_{ij}^{el}$) since the observed graph structure A contains noisy edges incurred by X and ϵ, which introduce inaccurate supervision.

More precisely, the regularized prediction $\hat{p}_{ij}^{reg}$ is defined as: $\hat{p}_{ij}^{reg} = \theta_1\hat{p}_{ij} + (1-\theta_1)s(\mathbf{X}_i,\mathbf{X}_j)$. Note that the feature similarity $s(\mathbf{X}_i,\mathbf{X}_j)$ is considered in the prediction of positive edges. The main idea is to penalize $\hat{p}_{ij}$ when $s(\mathbf{X}_i,\mathbf{X}_j)$ is high, as the edge between v_i and v_j is potentially noisy due to the influence of noisy X. For the regularized labels, we utilize the inferred ϵ_A, i.e., $\hat{p}_{ij}^{el}\in[0.9,1]$. In other words, when an edge is regarded as noisy (i.e., with a low $\hat{p}_{ij}^{el}$), its label is close to 0.9, while an edge considered clean (i.e., with a high $\hat{p}_{ij}^{el}$) has a label close to 1. This approach achieves a similar effect to label smoothing, enhancing the model robustness in the presence of noisy supervision. Note that θ_1 is a hyperparameter, and the 0.9 in the label regularization is selected following [25].

Modeling $p_{\theta_2}(X|\epsilon,Z_Y)$. The probability $p(X|\epsilon,Z_Y)$ indicates how well the noisy node feature X is reconstructed from the latent clean label Z_Y along with ϵ. Hence, we design the decoder θ_2 as a feature reconstruction model and aim to maximize $\log(p_{\theta_2}(X|\epsilon,Z_Y)))$ (or equivalently minimize the negative log likelihood). To do so, the decoder needs to rely on the information contained in Z_Y, which essentially encourages the value of Z_Y to be meaningful for the prediction process, i.e., generating X. It is worth noting that such learning objective is equivalent to minimizing $-\mathbb{E}_{\epsilon\sim q_{\phi_2}}\mathbb{E}_{Z_Y\sim q_{\phi_3}}\left[\log(p_{\theta_2}(X|\epsilon,Z_Y))\right]$ in Eqn. 4, which is implented as a feature reconstruction loss $\mathcal{L}_{\text{rec-feat}}$, where the decoder θ_2 is composed of an MLP that takes $CONCAT(\epsilon_X, Z_Y)$ as an input and outputs reconstructed node features. Note that the reparametrization trick [13] is used for sampling ϵ_X that follows the standard normal distribution.

Modeling $p_{\theta_3}(Y|X,A,Z_Y)$. The probability $p(Y|X,A,Z_Y)$ represents the transition relationship from the latent clean label Z_Y to the noisy label Y of an instance, i.e., how the label noise was generated [27]. For this reason, maximizing $\log(p_{\theta_3}(Y|X,A,Z_Y))$ (or equivalently minimizing the negative log likelihood) would let us discover the latent true label Z_Y from which the noisy label Y is generated given an instance, i.e., X and A. Hence, we aim to maximize the log likelihood, which is implemented as minimizing a node classification loss $\mathcal{L}_{\text{cls-dec}}$, i.e., the cross entropy loss. Specifically, the decoder θ_3 is composed of a GCN classifier that takes A and $CONCAT(X, Z_Y)$ as inputs, and outputs the prediction of Y, i.e., $\hat{\mathbf{Y}}_{\text{dec}} = \text{GCN}_{\theta_3}(\mathbf{X},\mathbf{A},\hat{\mathbf{Y}})\in\mathbb{R}^{N\times C}$. It is important to note that such a learning objective is equivalent to minimizing $-\mathbb{E}_{Z_Y\sim q_{\phi_3}}\left[\log(p_{\theta_3}(Y|X,A,Z_Y))\right]$ in Eqn. 4.

4.3.3 *Model Training.*

The overall learning objective can be written as follows and PRINGLE is trained to minimize the $\mathcal{L}_{\text{final}}$:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{cls-enc}} + \lambda_1\mathcal{L}_{\text{rec-edge}} + \lambda_2\mathcal{L}_{\text{hom}} + \lambda_3(\mathcal{L}_{\text{rec-feat}} + \mathcal{L}_{\text{cls-dec}} + \mathcal{L}_{\text{p}}), \tag{6}$$

where λ_1, λ_2, and λ_3 are the balancing coefficients.

Table 1: Node classification performance under synthetic feature-dependent graph-noise (FDGN).

Dataset	Setting	WSGNN	GraphGLOW	AirGNN	ProGNN	RSGNN	STABLE	EvenNet	NRGNN	RTGNN	PRINGLE
Cora	Clean	**86.2±0.1**	85.2±0.7	85.0±0.2	85.3±0.4	**86.2±0.5**	86.1±0.2	**86.2±0.0**	**86.2±0.2**	86.1±0.2	**86.2±0.7**
	FDGN-10%	80.7±0.3	79.7±0.2	79.7±0.5	79.6±0.7	81.9±0.3	82.2±0.7	80.7±0.7	81.0±0.5	81.8±0.3	**82.9±0.6**
	FDGN-30%	70.0±0.6	71.6±0.5	71.5±0.8	74.5±0.1	71.9±0.5	74.3±0.3	65.2±1.7	73.5±0.8	72.6±1.5	**78.2±0.3**
	FDGN-50%	55.9±1.1	59.6±0.1	56.2±0.8	66.4±0.4	59.9±0.5	62.8±2.4	47.1±1.8	61.9±1.4	60.9±0.4	**69.7±0.6**
Citeseer	Clean	76.6±0.6	76.5±1.0	71.5±0.2	72.6±0.5	75.8±0.4	74.6±0.6	76.4±0.5	75.0±1.3	76.1±0.4	**77.3±0.6**
	FDGN-10%	72.8±0.8	71.4±0.8	66.2±0.7	67.5±0.6	73.3±0.5	71.5±0.3	71.1±0.4	71.9±0.3	73.2±0.2	**74.3±0.9**
	FDGN-30%	63.3±0.7	60.6±0.2	58.0±0.4	61.0±0.2	63.9±0.5	62.5±1.4	61.2±0.6	62.5±0.7	64.2±1.9	**65.6±0.6**
	FDGN-50%	53.4±0.6	48.8±0.6	50.0±0.6	53.3±0.2	55.3±0.4	54.7±1.7	47.2±1.1	52.6±0.9	54.2±1.8	**59.0±1.8**
Photo	Clean	92.9±0.3	94.2±0.4	93.5±0.1	90.1±0.2	93.6±0.8	93.4±0.1	94.5±0.4	90.3±1.7	91.3±0.6	**94.8±0.3**
	FDGN-10%	83.9±1.8	92.1±0.8	87.3±0.9	84.3±0.1	92.1±0.2	92.2±0.1	92.6±0.0	84.3±1.3	89.4±0.5	**93.2±0.2**
	FDGN-30%	51.9±6.8	88.4±0.2	67.8±4.3	74.7±0.2	86.6±1.0	88.0±1.0	89.6±0.2	69.0±2.2	86.4±0.5	**90.5±0.4**
	FDGN-50%	31.9±5.6	85.4±0.6	57.8±0.7	48.9±0.5	75.6±2.6	80.2±1.8	84.6±0.4	57.5±1.8	79.2±0.3	**87.6±0.2**
Comp	Clean	83.1±3.1	91.3±0.9	83.4±1.2	83.9±0.8	91.1±0.1	90.2±0.2	90.1±0.2	87.5±1.0	87.3±1.0	**92.2±0.0**
	FDGN-10%	75.0±1.2	88.0±0.7	76.8±1.8	72.0±0.2	88.1±0.7	85.9±0.5	87.6±0.7	85.7±0.9	85.9±0.1	**89.8±0.2**
	FDGN-30%	48.5±5.8	84.9±0.4	59.2±0.9	66.9±0.8	81.7±0.2	80.4±1.0	84.8±0.5	74.8±3.5	77.0±1.5	**86.9±0.3**
	FDGN-50%	39.6±4.0	80.1±0.5	44.1±1.4	43.3±0.3	73.9±2.3	68.8±1.3	77.5±1.9	65.3±3.2	69.4±0.3	**82.2±0.4**

Table 2: Node classification performance under real-world FDGN.

	Auto		Garden	
Methods	Clean	+ FDGN	Clean	+ FDGN
WSGNN	71.8±4.3	57.7±1.3	87.4±0.2	77.6±0.8
GraphGLOW	77.9±1.2	59.4±0.8	88.5±0.9	78.1±1.5
AirGNN	69.5±0.8	53.9±0.1	78.3±1.5	66.1±1.7
ProGNN	63.2±0.2	48.6±0.3	78.7±0.1	73.0±0.4
RSGNN	69.5±0.4	56.8±0.9	83.3±1.2	76.2±0.5
STABLE	71.6±0.9	57.5±0.2	84.2±0.4	77.2±3.3
EvenNet	73.4±0.5	57.1±2.1	85.7±0.5	75.6±2.4
NRGNN	74.3±0.8	55.8±1.0	87.7±0.4	76.1±0.2
RTGNN	76.4±0.2	59.6±0.8	87.8±0.2	76.0±0.6
PRINGLE	**79.3±0.2**	**61.4±0.4**	**88.7±0.3**	**80.2±0.8**

Table 3: Link prediction performance under real-world FDGN.

	Auto		Garden	
Methods	Clean	+ FDGN	Clean	+ FDGN
WSGNN	81.8±0.1	69.1±0.6	84.7±0.2	84.6±0.7
GraphGLOW	86.2±0.3	**74.8±0.2**	90.2±0.5	90.1±0.4
AirGNN	60.2±0.2	57.9±0.4	62.0±0.1	58.2±0.5
ProGNN	74.8±0.3	56.7±0.5	83.5±0.6	83.3±0.5
RSGNN	87.2±0.8	65.0±0.2	91.2±0.4	91.2±0.5
STABLE	78.6±0.1	57.3±0.1	85.2±0.2	85.0±0.1
EvenNet	86.8±0.1	70.5±0.2	89.2±0.3	90.0±0.7
NRGNN	76.6±1.3	47.5±1.7	87.0±0.9	58.6±4.5
RTGNN	84.4±0.1	72.2±0.2	90.4±0.3	90.4±0.2
PRINGLE	**88.2±0.3**	73.6±0.6	**92.6±0.2**	**92.4±0.4**

5 EXPERIMENTS

Datasets. We evaluate PRINGLE and baselines on **four commonly used benchmark datasets** (i.e., Cora, Citeseer, Photo, and Computers) and **two newly introduced datasets** (i.e., Auto and Garden) based on Amazon review data [9, 22] to mimic FDGN on e-commerce systems.

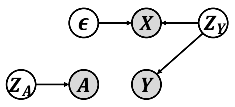
Experimental Details. We evaluated PRINGLE in both node classification and link prediction tasks, comparing it with noise-robust graph learning and generative graph learning methods. For a thorough evaluation, we create synthetic and real-world FDGN settings. We also account for independent structure/feature/label noise that are also prevalent in real-world applications, following [17, 19, 24]. Further details about the baselines, evaluation protocol, and implementation details can be found in Appendix A.1, A.2, and A.3, respectively.

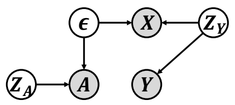
5.1 Quantitative Results

5.1.1 Under Synthetic Feature-Dependent Graph-Noise. We first evaluate PRINGLE under synthetic FDGN settings. Table 1 shows that PRINGLE consistently outperforms all baselines in FDGN scenarios, especially when noise levels are high. This superiority is attributed to the fact that PRINGLE captures the causal relationships involved in the DGP of FDGN, while the baselines overlook such relationships, leading to their model designs assuming the completeness of at least one data source. Additionally, PRINGLE performs well even in clean graph settings. We attribute this to the accurate inference of ϵ_A, which is utilized as the label regularization in calculating $\mathcal{L}_{\text{rec-edge}}$. Specifically, in Fig 6(a) in Sec 5.2.2, we observe that the $\hat{p}_{ij}^{el}$ values estimated from the clean graph tend to be close to 1, while those from the graph with FDGN are considerably smaller. Recall that the high value of $\hat{p}_{ij}^{el}$ indicates the model regards the edge (i, j) as a clean edge. This suggests that PRINGLE has the capability to adapt its model learning to the level of noise present in the input graph, resulting in superior performance on both clean and noisy graphs.

5.1.2 Under Real-world Feature-Dependent Graph-Noise. To investigate the robustness of PRINGLE under real-world noise scenarios, we newly design two benchmark graph datasets, i.e., Auto and Garden, where the node label is the product category, the node feature is bag-of-words representation of product reviews, and the edges indicate the co-purchase relationship between two products by the same user. To the best of our knowledge, this is the first work proposing new datasets for evaluating the noise-robust graph learning under realistic noise scenarios that are plausible in a real-world e-commerce system containing fraudsters. In Table 2 and 3, we observe that PRINGLE outperforms the baselines under FDGN caused by malicious actions of fraudsters on both the node classification and link prediction tasks. This indicates that PRINGLE works well not only under artificially generated noise, but also under noise scenarios that are plausible in real-world applications.

5.1.3 Under Independent Feature/Structure/Label Noise. We test how well PRINGLE handles independent noise in features, structure, and labels. See Appendix B.1 for details.


(a) Case 1


(b) Case 2

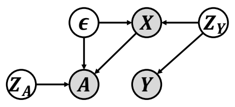

(c) Case 3

Figure 5: Graphical models of DGPs derived from FDGN.

Table 4: Ablation studies regarding various DGPs in Fig 5. In Case 3, the causal relationship $Y \leftarrow (X, A)$ is removed from the DGP of FDGN (See Fig 2(b)). In Case 2, $A \leftarrow X$ is additionally removed. In Case 1, $A \leftarrow \epsilon$ is additionally removed, which is equivalent to the DGP of CGN (See Fig 2(a)). Cora and Citeseer datasets are used to evaluate the node classification performance.

Dataset	Setting	(a) Case 1	(b) Case 2	(c) Case 3	PRINGLE
Cora	Clean	84.6±0.4	84.8±0.4	**86.2±0.2**	**86.2±0.7**
	FDGN-10%	77.4±0.3	77.3±0.3	**83.2±0.3**	82.9±0.6
	FDGN-30%	68.3±0.4	68.5±0.2	77.3±0.4	**78.2±0.3**
	FDGN-50%	55.2±0.2	56.1±0.3	68.7±0.3	**69.7±0.6**
Citeseer	Clean	76.7±0.9	76.8±0.8	76.5±0.9	**77.3±0.6**
	FDGN-10%	69.5±0.3	69.5±0.4	73.2±0.1	**74.3±0.9**
	FDGN-30%	57.2±1.1	57.7±0.5	65.5±0.7	**65.6±0.6**
	FDGN-50%	49.2±0.5	48.7±0.2	57.6±2.5	**59.0±1.8**

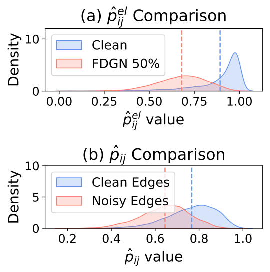


Figure 6: (a) Distribution of $\hat{p}_{ij}^{el}$ values of clean and FDGN-50%. (b) Distribution of $\hat{p}_{ij}$ values of clean and noisy edges under FDGN-50%. Dashed lines are averages. Cora dataset is used.

5.2 Model Analyses

5.2.1 Ablation Studies. To emphasize the importance of directly capturing the causal relationships among variables in the DGP of FDGN, i.e., $Y \leftarrow (X, A)$, $A \leftarrow X$, and $A \leftarrow \epsilon$, we remove them one by one from the graphical model of FDGN (See Fig 2(b), and then design deep generative models based on the DGPs in a similar manner to PRINGLE. The graphical models of the derived DGPs are illustrated in Fig 5. In Table 4, we observe that as more causal relationships are removed from the DGP of FDGN, the node classification performance decreases. Below, we offer explanations for this observation from the perspective of model derivation.

1) When removing the causal relationships $Y \leftarrow (X, A)$, i.e., Fig 5(c), the loss term $-\mathbb{E}_{Z_Y \sim q_{\phi_3}}\left[\log(p_{\theta_3}(Y|X, A, Z_Y))\right]$ can be simplified to $-\mathbb{E}_{Z_Y \sim q_{\phi_3}}\left[\log(p_{\theta_3}(Y|Z_Y))\right]$. This simplification hinders the accurate modeling of the label transition relationship from Z_Y to the noisy label Y, resulting in a degradation of model performance under FDGN.

2) Additionally, when eliminating the causal relationship $A \leftarrow X$, i.e., Fig 5(b), the inference of Z_A and Z_Y is simplified as follows: $q_{\phi_1}(Z_A|X, A)$ to $q_{\phi_1}(Z_A|A)$ and $q_{\phi_3}(Z_Y|X, A)$ to $q_{\phi_3}(Z_Y|X)$. Furthermore, the loss term $-\mathbb{E}_{Z_A \sim q_{\phi_1}}\mathbb{E}_{\epsilon \sim q_{\phi_2}}\left[\log(p_{\theta_1}(A|X, \epsilon, Z_A))\right]$ is also simplified to $-\mathbb{E}_{Z_A \sim q_{\phi_1}}\mathbb{E}_{\epsilon \sim q_{\phi_2}}\left[\log(p_{\theta_1}(A|\epsilon, Z_A))\right]$. These simplifications significantly hinder the accurate inference of Z_A and Z_Y, resulting in a notable performance degradation.

3) Furthermore, when removing the causal relationship $A \leftarrow \epsilon$, i.e., Fig 5(a), the loss term $-\mathbb{E}_{Z_A \sim q_{\phi_1}}\mathbb{E}_{\epsilon \sim q_{\phi_2}}\left[\log(p_{\theta_1}(A|\epsilon, Z_A))\right]$ is simplified to $-\mathbb{E}_{Z_A \sim q_{\phi_1}}\mathbb{E}_{\epsilon \sim q_{\phi_2}}\left[\log(p_{\theta_1}(A|Z_A))\right]$. This simplification hinders the robustness of the inferred Z_A, since the simplified loss excludes label regularization from the model training process, ultimately resulting in performance degradation.

5.2.2 Analysis of the inferred Z_A and ϵ_A. In this subsection, we qualitatively analyze how well PRINGLE infers the latent variable ϵ_A and Z_A. In Fig 6(a), we conducted an analysis of the inference of ϵ_A by comparing the distribution of $\hat{p}_{ij}^{el}$ values estimated during the training of PRINGLE on clean and noisy graphs (FDGN-50%). We observe that $\hat{p}_{ij}^{el}$ values estimated from the clean graph tend to be close to 1, while those from the graph with FDGN are considerably smaller. This observation suggests that the inference of ϵ_A was accurate, as the high values of $\hat{p}_{ij}^{el}$ indicate that the model recognizes the edge (i, j) as a clean edge.

In Fig 6(b), we analyze the inference of Z_A by comparing the distribution of $\hat{p}_{ij}$ values, which constitute the estimated latent graph structure $\hat{\mathbf{A}}$, between noisy edges and the original clean edges. It is evident that the estimated edge probabilities $\hat{p}_{ij}$ for noisy edges are predominantly assigned smaller values, while those for clean edges tend to be assigned larger values. This observation illustrates that PRINGLE effectively mitigates the impact of noisy edges during the message-passing process, thereby enhancing its robustness in the presence of noisy graph structure. This achievement can be attributed to the label regularization effect achieved through the accurate inference of ϵ_A. Specifically, as the observed graph structure contains noisy edges, the inaccurate supervision for $\mathcal{L}_{\text{rec-edge}}$ impedes the distinction between noisy edges and the original clean edges in terms of edge probability values $\hat{p}_{ij}$. However, the label regularization technique proves crucial for alleviating this issue, benefitting from the accurate inference of ϵ_A.

6 CONCLUSION

In this paper, we discover practical limitations of conventional graph noise in terms of node features, i.e., the noise in node features is independent of the graph structure or node label. To mitigate limitations of the conventional graph noise assumption, we newly introduce a more realistic graph noise scenario called feature-dependent graph-noise (FDGN), and present a deep generative model that effectively captures the causal relationships among variables in the DGP of FDGN. Our proposed method, PRINGLE, consistently outperforms baselines in both node classification and link prediction tasks. We evaluate PRINGLE on commonly used benchmark datasets and newly introduced real-world graph datasets that simulate FDGN in e-commerce systmes, which is expected to foster practical research in noise-robust graph learning.

ACKNOWLEDGEMENTS

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00077, No.2022-0-00157).

REFERENCES

[1] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. 2017. A closer look at memorization in deep networks. In *International conference on machine learning*. PMLR, 233–242.

[2] Antonin Berthon, Bo Han, Gang Niu, Tongliang Liu, and Masashi Sugiyama. 2021. Confidence scores make instance-dependent label-noise learning possible. In *International conference on machine learning*. PMLR, 825–836.

[3] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association* 112, 518 (2017), 859–877.

[4] Yoonhyuk Choi, Jiho Choi, Taewook Ko, Hyungho Byun, and Chong-Kwon Kim. 2022. Finding Heterophilic Neighbors via Confidence-based Subgraph Matching for Semi-supervised Node Classification. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 283–292.

[5] Enyan Dai, Charu Aggarwal, and Suhang Wang. 2021. 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*. 227–236.

[6] Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. 2022. Towards Robust Graph Neural Networks for Noisy Graphs with Sparse Labels. *WSDM* (2022).

[7] Pantelis Elinas, Edwin V Bonilla, and Louis Tiao. 2020. Variational inference for graph convolutional networks in the absence of graph data and adversarial settings. *Advances in Neural Information Processing Systems* 33 (2020), 18648–18660.

[8] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. 2021. SLAPS: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 22667–22681.

[9] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.

[10] Ahmet Iscen, Jack Valmadre, Anurag Arnab, and Cordelia Schmid. 2022. Learning with neighbor consistency for noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4672–4681.

[11] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 66–74.

[12] Wei Jin, Tong Zhao, Jiayuan Ding, Yozen Liu, Jiliang Tang, and Neil Shah. 2022. Empowering graph representation learning with test-time graph transformation. *arXiv preprint arXiv:2210.03561* (2022).

[13] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[14] Danning Lao, Xinyu Yang, Qitian Wu, and Junchi Yan. 2022. Variational inference for training graph neural networks in low-data regime through joint structure-label estimation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 824–834.

[15] Runlin Lei, Zhen Wang, Yaliang Li, Bolin Ding, and Zhewei Wei. 2022. Evennet: Ignoring odd-hop neighbors improves robustness of graph neural networks. *arXiv preprint arXiv:2205.13892* (2022).

[16] Junnan Li, Richard Socher, and Steven CH Hoi. 2020. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394* (2020).

[17] Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. Reliable Representations Make A Stronger Defender: Unsupervised Structure Refinement for Robust GNN. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 925–935.

[18] Nian Liu, Xiao Wang, Lingfei Wu, Yu Chen, Xiaojie Guo, and Chuan Shi. 2022. Compact Graph Structure Learning via Mutual Information Compression. In *Proceedings of the ACM Web Conference 2022*. 1601–1610.

[19] Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. 2021. Graph neural networks with adaptive residual. *Advances in Neural Information Processing Systems* 34 (2021), 9720–9733.

[20] Jiaqi Ma, Weijing Tang, Ji Zhu, and Qiaozhu Mei. 2019. A flexible generative framework for graph-based semi-supervised learning. *Advances in Neural Information Processing Systems* 32 (2019).

[21] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. 2021. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[22] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.

[23] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.

[24] Siyi Qian, Haochao Ying, Renjun Hu, Jingbo Zhou, Jintai Chen, Danny Z Chen, and Jian Wu. 2023. Robust Training of Graph Neural Networks via Noise Governance. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 607–615.

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[26] Yijun Tian, Chuxu Zhang, Zhichun Guo, Xiangliang Zhang, and Nitesh Chawla. 2023. Learning MLPs on Graphs: A Unified View of Effectiveness, Robustness, and Efficiency. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Cs3r5KLdoj

[27] Yu Yao, Tongliang Liu, Mingming Gong, Bo Han, Gang Niu, and Kun Zhang. 2021. Instance-dependent label-noise learning under a structural causal model. *Advances in Neural Information Processing Systems* 34 (2021), 4409–4420.

[28] Wentao Zhao, Qitian Wu, Chenxiao Yang, and Junchi Yan. 2023. GraphGLOW: Universal and Generalizable Structure Learning for Graph Neural Networks. *arXiv preprint arXiv:2306.11264* (2023).

A DETAILS ON EXPERIMENTAL SETTINGS

A.1 Baselines

We compare PRINGLE with a wide range of noise-robust graph learning methods, which includes feature noise-robust grah learning methods (i.e., AirGNN [19]), structure-noise robust graph learning methods (i.e., ProGNN [11], RSGNN [6], STABLE [17] and EvenNet [15]), and label noise-robust graph learning methods (i.e., NRGNN [5] and RTGNN [24]). We also consider WSGNN [14] and GraphGLOW [28] that are generative graph learning methods utilizing variational inference technique. The publicly available implementations of baselines can be found at the following URLs:

- **WSGNN** [14] : https://github.com/Thinklab-SJTU/WSGNN
- **GraphGLOW** [28] : https://github.com/WtaoZhao/GraphGLOW
- **AirGNN** [19] : https://github.com/lxiaorui/AirGNN
- **ProGNN** [11] : https://github.com/ChandlerBang/Pro-GNN
- **RSGNN** [6] : https://github.com/EnyanDai/RSGNN
- **STABLE** [17] : https://github.com/likuanppd/STABLE
- **EvenNet** [15] : https://github.com/Leirunlin/EvenNet
- **NRGNN** [6] : https://github.com/EnyanDai/NRGNN
- **RTGNN** [6] : https://github.com/GhostQ99/RobustTrainingGNN

A.2 Evaluation Protocol

We mainly compare the robustness of PRINGLE and the baselines under both the synthetic and real-world feature-dependent graph-noise (FDGN). Additionally, we consider independent feature/structure/label noise, i.e., random feature noise, random structure noise, uniform label noise, and pair label noise following existing works [6, 17, 19, 24].

We conduct both the node classification and link prediction tasks. For node classification, we perform a random split of the nodes, dividing them into a 1:1:8 ratio for training, validation, and testing nodes. Once a model is trained on the training nodes, we use the model to predict the labels of the test nodes. Regarding link prediction, we partition the provided edges into a 7:3 ratio for training and testing edges. Additionally, we generate random negatives that are selected randomly from pairs that are not directly linked in the original graphs. After mode learning with the training edges, we predict the likelihood of the existence of each edge. This prediction is based on a dot-product or cosine similarity calculation between node pairs of test edges and their corresponding negative edges. To evaluate performance, we use Accuracy as the metric for node classification and Area Under the Curve (AUC) for link prediction.

A.3 Implementation Details

For each experiment, we report the average performance of 3 runs with standard deviations. For all baselines, we use the publicly

Table 5: Hyperparameter settings on PRINGLE for Table 1.

Dataset	Setting	lr	λ_1	λ_2	θ_1	k	γ
Cora	Clean	0.01	0.003	0.003	0.1	300	1
	FDGN-10%	0.005	0.003	0.003	0.2	50	1
	FDGN-30%	0.001	0.003	0.003	0.2	100	1
	FDGN-50%	0.0005	30	0.003	0.3	50	1
Citeseer	Clean	0.0005	0.003	0.3	0.1	50	0
	FDGN-10%	0.005	0.3	0.003	0.3	10	0
	FDGN-30%	0.001	0.003	0.003	0.1	300	1
	FDGN-50%	0.001	0.003	0.003	0.1	300	1
Photo	Clean	0.01	0.03	0.3	0.1	10	0
	FDGN-10%	0.0005	0.03	0.3	0.1	10	0
	FDGN-30%	0.001	3	0.03	0.1	10	0
	FDGN-50%	0.0005	30	0.03	0.1	10	0
Comp	Clean	0.01	30	0.03	0.1	10	0
	FDGN-10%	0.01	0.3	0.03	0.1	10	0
	FDGN-30%	0.01	0.003	0.003	0.1	10	0
	FDGN-50%	0.0005	0.003	0.03	0.1	10	0

available implementations and follow the implementation details presented in their original papers. For PRINGLE, we report the details of hyperparameter settings in Table 5.

B ADDITIONAL EXPERIMENTAL RESULTS

B.1 Under Independent Feature/Structure/Label Noise

In this subsection, we further evaluate the robustness of PRINGLE under independent feature/structure/label noise settings. In this setup, each type of noise occurs independently and does not affect the occurrence of the others. To this end, we generate three types of noise: random feature noise, random structure noise, and random label noise.

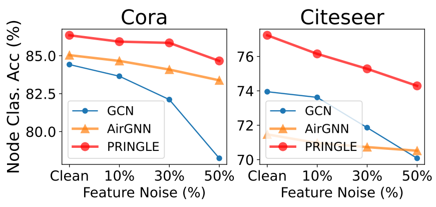


Figure 7: Node classification performance of baselines and PRINGLE on Cora and Citeseer dataset under independent feature noise (noise rate from 0% to 50%).

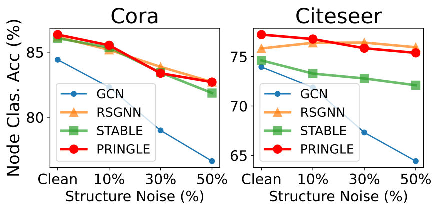


Figure 8: Node classification performance of baselines and PRINGLE on Cora and Citeseer dataset under independent structure noise (noise rate from 0% to 50%).

Evaluating robustness under independent feature noise. In this setting, we evaluate the models on a graph containing only the feature noise, i.e., random feature noise. In Fig 7, PRINGLE consistently outperforms the feature noise-robust graph learning method (i.e., AirGNN) under independent feature noise. We attribute the robustness of PRINGLE under independent feature noise to the graph structure learning module that accurately infers the latent graph structure Z_A. The utilization of abundant local neighborhoods acquired through the inference of Z_A enables effective smoothing for nodes with noisy features, leveraging the information within these neighborhoods.

Evaluating robustness under independent structure noise. In this setting, we evaluate the models on a graph containing only the structure noise, i.e., random structure noise. In Fig 8, Under the influence of independent structure noise, PRINGLE maintains a consistently competitive performance when compared to other structure noise-robust graph learning methods, namely RSGNN and STABLE. We attribute the effectiveness of PRINGLE under independent structure noise to inferring the robust latent clean graph structure. In other words, the inference of the latent clean graph structure Z_A assigns greater weights to latent clean edges and lower weights to observed noisy edges by employing regularizations on both the edge predictions and labels, thereby mitigating structural noise.

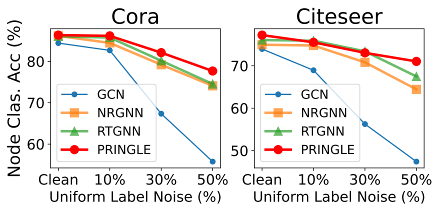


Figure 9: Node classification performance of baselines and PRINGLE on Cora and Citeseer dataset under independent uniform label noise (noise rate from 0% to 50%).

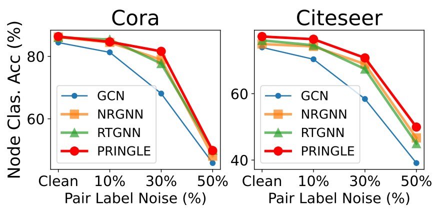


Figure 10: Node classification performance of baselines and PRINGLE on Cora and Citeseer dataset under independent pair label noise (noise rate from 0% to 50%).

Evaluating robustness under independent label noise In this setting, we evaluate the models on a graph containing only the label noise, i.e., uniform label noise and pair label noise. In Fig 9 and 10, PRINGLE demonstrates consistent superiority or competitive performance compared to label noise-robust graph learning methods, namely NRGNN and RTGNN, in the presence of independent label noise. We argue that the effectiveness of PRINGLE stems from the accurate inference of the latent clean structure. Specifically, the inferred latent node label Z_Y is regularized using the inferred latent structure Z_A to meet the homophily assumption (i.e., $\mathcal{L}_{\text{hom}}$). Leveraging the clean neighbor structure, this regularization technique has been demonstrated to effectively address noisy labels [10].