Noise-Resilient Unsupervised Graph Representation Learning via Multi-Hop Feature Quality Estimation

Shiyuan Li*
Guangxi University
Nanning, China
shiy.li@alu.gxu.edu.cn

Yixin Liu*
Griffith University
Gold Coast, Australia
yixin.liu@griffith.edu.au

Qingfeng Chen†
Guangxi University
Nanning, China
qingfeng@gxu.edu.cn

Geoffrey I. Webb
Monash University
Melbourne, Australia
geoff.webb@monash.edu

Shirui Pan
Griffith University
Gold Coast, Australia
s.pan@griffith.edu.au

ABSTRACT

Unsupervised graph representation learning (UGRL) based on graph neural networks (GNNs), has received increasing attention owing to its efficacy in handling graph-structured data. However, existing UGRL methods ideally assume that the node features are noise-free, which makes them fail to distinguish between useful information and noise when applied to real data with noisy features, thus affecting the quality of learned representations. This urges us to take node noisy features into account in real-world UGRL. With empirical analysis, we reveal that feature propagation, the essential operation in GNNs, acts as a "double-edged sword" in handling noisy features - it can both denoise and diffuse noise, leading to varying feature quality across nodes, even within the same node at different hops. Building on this insight, we propose a novel UGRL method based on Multi-hop feature Quality Estimation (MQE for short). Unlike most UGRL models that directly utilize propagation-based GNNs to generate representations, our approach aims to learn representations through estimating the quality of propagated features at different hops. Specifically, we introduce a Gaussian model that utilizes a learnable "meta-representation" as a condition to estimate the expectation and variance of multi-hop propagated features via neural networks. In this way, the "meta representation" captures the semantic and structural information underlying multiple propagated features but is naturally less susceptible to interference by noise, thereby serving as high-quality node representations beneficial for downstream tasks. Extensive experiments on multiple real-world datasets demonstrate that MQE in learning reliable node representations in scenarios with diverse types of feature noise.

*Both authors contributed equally to this research.
†Corresponding author.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; • **Mathematics of computing → Graph algorithms**.

KEYWORDS

Unsupervised Learning, Graph Representation Learning, Feature Quality, Gaussian Model

1 INTRODUCTION

Unsupervised graph representation learning (UGRL) aims to automatically extract low-dimensional vector representations from graph-structured data, eliminating the need for manual annotation [10, 16]. The learned representations can be utilized for diverse downstream graph learning tasks, including node classification, node clustering, and graph classification [13, 27, 32, 47, 50, 55]. Given their powerful capability to model graph data, graph neural networks (GNNs) [18, 23, 33, 54] have become the de facto backbone models for various UGRL approaches [28, 48, 53, 60]. In recent years, UGRL has found applications across several domains, such as drug discovery, fraud detection, recommendation systems, and traffic forecasting [7, 42, 43, 45].

The majority of UGRL models rely on the fundamental assumption that the observed node features are uncontaminated, thereby enabling the utilization of these features to furnish ample self-supervised signals for model training [13, 61]. Unfortunately, such an assumption is often invalid in real-world scenarios, as graph data extracted from complex systems frequently contains noisy, incomplete, and even erroneous features [2, 8, 21, 22, 24, 57]. For example, considering privacy issues, users in many social networks may provide blank or even false information. Moreover, textual features may contain many spelling errors or informal expressions [14, 49]. As illustrated in Fig. 1 (a), (b) and (e), the presence of noise can significantly degrade both the quality and distribution of the original features, resulting in indistinguishable node features. As a result,
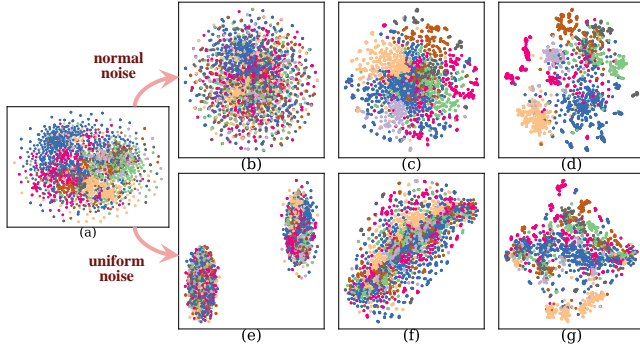
Figure 1: t-SNE [40] visualization of propagated features by symmetric normalized adjacency matrix of Cora dataset. (a) original features; (b)/(e) features after perturbation by noise; (c),(f)/(d),(g) noisy features after 2/16-step propagation.
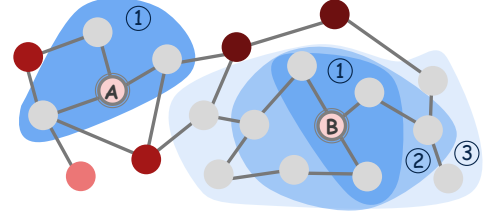

Figure 2: A toy example to illustrate the optimal propagation steps for different nodes. The color darkness of each node indicates its noise intensity (clean-lightly noisy-heavily noisy). For node A, a 1-step propagation that aggregates its 1-hop messages is beneficial for feature denoising. For node B, a larger number of propagation steps (e.g., 3) is preferred to generate a reliable representation by aggregating more neighboring information.

these practical issues can lead to imperfect graph data characterized by noisy features, which poses a challenge to existing UGRL models. Specifically, contrastive methods often depend on prior-based augmentation to produce different views and construct positive and negative samples [60]. However, noisy features may obscure the boundaries between positive and negative samples, making it challenging for the model to accurately differentiate between them. Reconstruction-based methods, e.g., GraphMAE [13], usually involve the reconstruction of the original features, which can lead to model fitting to noisy information. Therefore, in the absence of reliable supervision signals, the noisy features inevitably prevent us from training an expressive UGRL model on real-world graph data. To remedy this shortcoming, a natural research question is raised: ***How can we develop a robust UGRL approach that effectively addresses the presence of noisy features?***

To answer this question, this paper first conducts a comprehensive analysis to investigate the performance of UGRL models when learning from noisy features. Through empirical discussions, we find that *propagation, the fundamental operation for information aggregation in GNNs, is a double-edged sword in the case of noisy features*. On the one hand, under the assumption that noises are often high-frequency graph signals, propagation functions as a denoising mechanism in handling noisy graph data [30]. From the spectral perspective, propagation operation is essentially equivalent to low-pass filtering in graph signal processing, which effectively suppresses the high-frequency portion of the message and thus reduces the effect of noise in signal passing. In this case, propagation is not only a way of information broadcasting, but also a signal processing filter that enhances the robustness of GNN-based UGRL models against the noisy feature issue. As illustrated in Fig. 1 (c) and (d), propagation operations with different steps make noisy features more discernible, with features of the same category forming tightly knit clusters.

On the other hand, propagation is not always the perfect solution for handling graphs with noisy features. As shown in Fig. 1 (e), (f), and (g), for uniform noise, the propagation operation cannot remove the noise stably. In this case, the noisy signals will spread to the surrounding nodes along with the propagation process, affecting the representation quality of nodes with clean or

slightly noisy features. Moreover, due to the uneven distribution of noise node positions and noise levels, the propagation steps to obtain high-quality representation for each node, which strike a balance between feature denoising and noise spreading, can vary significantly. Fig. 2 provides an example to illustrate this situation, where the representation quality at a fixed propagation step can be diverse across different nodes. More severely, in the scenario of UGRL, it is non-trivial to customize the best propagation step for each node without the guidance of ground-truth labels. Based on the above findings, an immediate core question is: ***How to measure the representation quality obtained from different steps of propagation to derive higher-quality representations?***

To answer the question, we propose a robust and efficient UGRL model termed **M**ulti-hop feature **Q**uality **E**stimation (MQE for short). Our theme is to adaptively learn the optimal representations of each node by explicitly estimating the quality of propagated features at different hops. More specifically, to exploit the inherited denoising capability of long-range propagation, we first perform multi-hop propagation on the original features as a prior observation. Then, we introduce a Gaussian model to estimate the quality of propagated features by modeling the truth signals and noise information under different hops. In the process of feature quality estimation, we introduce a learnable matrix $\mathbf{Z}$ termed "meta representation" to indicate the essential information of each node for different hops. The meta representation $\mathbf{Z}$ can serve as the high-quality node representations learned by MQE, since it summarizes the most informative signals from propagated node features at different hops. Apart from learning reliable node representations, MQE can also quantize the noise intensity of node features during quality estimation, which provides potential explanation and analysis for graph data. Notably, MQE is different from the vast majority of UGRL approaches: rather than generating representations through propagation or GNNs, we attempt to estimate and fit different propagated features through learnable representations and neural networks. This characteristic enables MQE to leverage the denoising merit multi-hop propagation while alleviating the side effect of noise spreading. Overall, the contributions of this paper are as follows:

- **Problem.** We conduct comprehensive analyses to study the impact of noisy features on UGRL models, which provides an

in-depth understanding of the research problem and model design.

- **Method.** We propose a novel method termed MQE, which aims to estimate the quality of propagated features and adaptively learn high-quality representations from multi-hop information.
- **Experiments.** We conduct extensive experiments on graph data with various types of feature noises. The experimental results show that MQE effectively learns reliable node representations and estimates the noise intensity of features.

2 NOTATIONS & PRELIMINARY

2.1 Notation

Given an attribute graph denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \cdots, v_n\}$ is the set of with $|\mathcal{V}| = n$ nodes and $\mathcal{E}$ is the set of edges. The adjacency matrix of the $\mathcal{G}$ is denoted as $\mathbf{A}$, where the i, j-th entry $a_{ij} = 1$ if and only if the i-th and j-th nodes are connected (i.e. $(v_i, v_j) \in \mathcal{E}$), otherwise $\mathbf{a}_{ij} = 0$. The symmetric normalization of the adjacency matrix is denoted by $\hat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \widetilde{\mathbf{D}}^{-\frac{1}{2}}$, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ represents the adjacency matrix of the undirected graph with the addition of the self-loops and $\widetilde{\mathbf{D}}$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$. We denote the neighbor set of node v_i as $\mathcal{N}_i = \{v_j | (v_i, v_j) \in \mathcal{E}\}$. The attributes of nodes $\mathcal{V}$ can be represented by the node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, where d is the feature dimension and the i-th row $\mathbf{x}_i$ represents the feature vector of the i-th node v_i.

2.2 Problem Definition

Unsupervised graph representation learning (UGRL) aims to learn a mapping function $\mathcal{F}$ to obtain high-quality representations for all nodes in $\mathcal{G}$: $\mathcal{F}(\mathbf{A}, \mathbf{X}) \rightarrow \mathbf{Z} \in \mathbb{R}^{n \times f}$, where $\mathbf{Z}$ is the learned representation matrix, with each row $\mathbf{z}_i$ indicating the f-dimensional $(f \ll d)$ representation vector of each node. These representations can be used for numerous downstream tasks, such as node classification and node clustering.

In mainstream UGRL settings, ones usually assume that the observed node features in the graph data are clean and informative enough to indicate the node contexts. However, in certain real-world scenarios, the feature vectors used for model training often contain noise, which can significantly impact the quality of learned representations and, consequently, the performance of downstream tasks. To address this real-world challenge, our paper specifically focuses on the UGRL problem on graphs with noisy features.

Formally, the graph feature matrix with noise can be defined by $\mathbf{X} = \widetilde{\mathbf{X}} + \mathbf{\Psi}$, where $\widetilde{\mathbf{X}}$ is the clean feature data (a.k.a. true signal in signal processing [30]) and $\mathbf{\Psi}$ is the noise. Regarding the degree of noise within each node feature vector can be different, we further introduce *noise intensity* to indicate the strength of noise of different nodes. To quantize noise intensity s_i for node v_i, we can employ the modified L2 distance between true signal $\widetilde{\mathbf{x}}_i$ and observed signal $\mathbf{x}_i$, i.e., $s_i = \sqrt{\frac{1}{d} \sum_{j=1}^{d} (x_{ij} - \widetilde{x}_{ij})^2}$. In our target scenario, a powerful UGRL model $\mathcal{F}(\mathbf{A}, \mathbf{X}) \rightarrow \mathbf{Z}$ is expected to learn high-quality representations $\mathbf{Z}$ from graph data with noisy features $\mathbf{X}$. Apart from representation learning, the proposed UGRL method, MQE, can also estimate the noise intensity s_i for each node.

3 DESIGN MOTIVATION AND ANALYSIS

3.1 Can Propagation Alleviate Noisy Feature Problem in UGRL?

Propagation is a fundamental operation in GNNs for inter-node information communication along edges in graph-structured data. Since propagation has been proven to be effective in graph signal denoising on supervised GNNs [25, 26, 30], in the context of unsupervised scenarios, we are also curious about ***whether propagation helps handle the noisy feature problem in UGRL***.

Theoretical Justification. From the perspective of graph signal processing [31], the mainstream GNNs (e.g., GCN [17] and SGC [44]) can be regarded as low-pass filters for graph signals (i.e., node features). Specifically, Theorem 3 in [30] indicates that by multiplying the graph signals with the propagation matrix, the low-frequency components within the graph signals are typically retained; Conversely, the high-frequency counterparts tend to be attenuated or smoothed out. Considering the basic assumption in signal processing that the true signals are usually low-frequency, the propagation operation in GNNs can inherently suppress the noise signals in features. Theoretically speaking, even in UGRL scenarios where supervised signals are absent, this principle still holds true.

Intuitive Solution. Given that propagation theoretically aids in feature denoising, a straightforward and effective approach to accentuate the denoising effect is to amplify the number of propagation steps [25]. From the perspectives of both feature and structure, a larger propagation step can generally mitigate the noisy feature problem. On the feature side, a larger propagation step allows the model to consider a wider range of neighboring nodes, providing richer contextual knowledge for information aggregation while enabling noise filtering more effectively. On the structural side, such long-range propagation allows for communication between distant nodes, leveraging the existing graph structure knowledge more effectively in denoising. Preliminary visualization results in Fig. 1 (b)-(d) verify that a larger propagation step can lead to more discriminative representation distribution on graph data with normal noise. However, Fig. 1 (e)-(g) also illustrates that propagation is not always effective. Conversely, it may lead to noise diffusion thus affecting the quality of the representation of clean nodes.

Empirical Analysis. Based on the above discussion, we find that propagation may not always favor feature denoising. Therefore, to further quantify the impact of propagation more concretely, a natural question is: ***Would directly increasing the number of propagation steps in existing UGRL models suffice to learn informative representations from noisy features?*** To answer this question, with node classification as the downstream task, we assess the representation capability of prominent UGRL methods (specifically, DGI [41] and GRACE [60]) using both default (2), moderate (4) and large (16) propagation steps, across various noisy feature scenarios on the Cora dataset. To prevent the performance degradation [51] of GNNs due to unnecessary transformation operations affecting the analysis results, SGC [44], a GNN with fix-step transformation, is uniformly employed as the backbone network for both DGI and GRACE.

The experimental results are demonstrated in Fig. 3. We can summarize the following observations. 1) With the growth of noise level, the performance of DGI and GRACE significantly decreases.
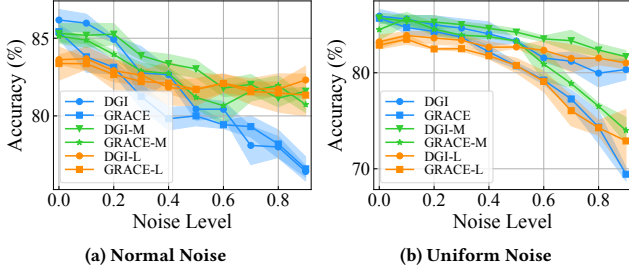
Figure 3: The performance of UGRL models with default propagation step, moderate propagation (with suffix "M" and larger propagation step (with suffix "L") on the Cora dataset with different noise levels and types.

2) For the scenarios with larger noise levels (i.e. > 0.5), the models with larger propagation steps always perform better than their default counterparts. 3) For situations with clean or slightly noisy features, the gain from increasing the propagation step is small or even negative. 4) In uniform noise scenarios, increasing the propagation step only may further diffuse the noise, making the model performance degraded.

Discussion. From the above empirical analysis, we can conclude that propagation is a double-edged sword - it can both denoise and diffuse noise. Specifically, increasing the propagation step is beneficial for UGRL models when the noise level is relatively high. Nevertheless, this is not a perfect solution, regarding the sub-optimal performance of large-step models in several scenarios. Therefore, a straightforward solution is to choose an optimal propagation step size to balance denoising and noise spreading.

3.2 Is there a Fixed Optimal Propagation Step?

In the above subsection, we delved into the effects of employing a larger propagation step to alleviate the issue of noisy features in UGRL. In this subsection, we discuss the impact of propagation from a more nuanced viewpoint: ***Is there a fixed optimal propagation step for every node to acquire high-quality representation?***

Empirical Analysis. To answer the aforementioned questions, we conducted an empirical experiment on Cora dataset with feature noise levels of 0.5. Specifically, we vary the propagation step of the SGC encoder in DGI and count the "optimal propagation step" for each node. Here, the term "optimal propagation step" refers to the specific propagation step that yields the highest average accuracy across multiple trials for the respective node.

The statistics of the distribution of optimal propagation steps are shown in Fig. 4. Our observations are given as follows. 1) The optimal propagation steps for different nodes can vary significantly. Specifically, for the majority of nodes, a larger propagation step usually yields better performance; however, there is still a small fraction of nodes that benefit from a smaller propagation step (≤ 2). 2) Selecting a propagation step that universally benefits all nodes is difficult, as the distribution of optimal steps remains uniform. 3) The distribution of optimal steps may vary depending on the type of feature noises.

Discussion. With empirical verification, we identify the challenge of selecting an optimal propagation step that consistently enhances
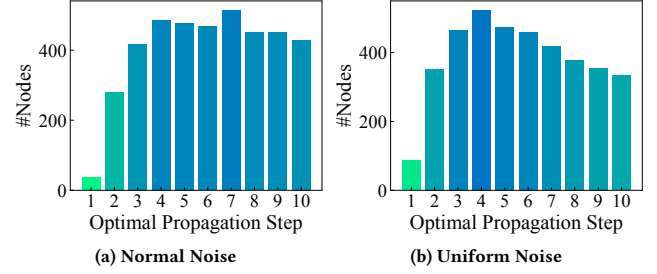

Figure 4: The distribution of optimal propagation step for each node on Cora dataset with different noise types.

the representation quality for all nodes. ***Why can larger propagation steps degrade the representation quality for several nodes?*** Regarding the presence of noisy features, we attribute such degradation to the spreading of noise through information propagation. Specifically, when a node's noisy features are difficult to denoise using low-pass filtering, it can greatly diminish the representation quality of its neighboring nodes. In this case, a node close to heavily noisy nodes may prefer a smaller propagation step to maintain the purity of its representations.

Furthermore, because nodes vary in their positions and feature noise intensity, the optimal propagation steps can vary across different nodes. This phenomenon is illustrated with a toy example in Fig. 2. In this example, node B requires a larger propagation step to access more beneficial information, whereas node A may receive more noisy information when using a large propagation step. Such diversity, inherently, hinders us from using UGRL models with a singular propagation step to acquire high-quality representation for all nodes.

Challenge. The above deduction exposes that the quality of representation can undergo significant alterations based on the number of propagation steps utilized. In order to learn reliable representations for all nodes, a potential way is to fuse the representations acquired from multi-hop propagation [1, 52, 58]. Nevertheless, unlike the supervised models [1, 52, 58] that can learn to aggregate multi-hop information automatically, in UGRL models, it is nontrivial to decide the contributions of representations from different hops during aggregation without the guidance of labels. Moreover, multi-hop aggregation will also more or less allow noise information to mix into the final representation. In this case, a more promising solution is to explicitly estimate the quality of propagated features, and subsequently extract reliable representations from the high-quality ones.

Summary: With comprehensive analyses, we conclude that propagation serves as a double-edged sword in addressing the noisy feature problem in UGRL, embodying the trade-off between spectral graph signal denoising and spatial noise message spreading. Moreover, the diverse distribution of noisy nodes results in variations in the quality of node representations across different propagation steps. This diversity underscores the challenge of obtaining reliable representations through a fixed propagation step. In other words, the optimal steps vary among nodes rather than remaining fixed.
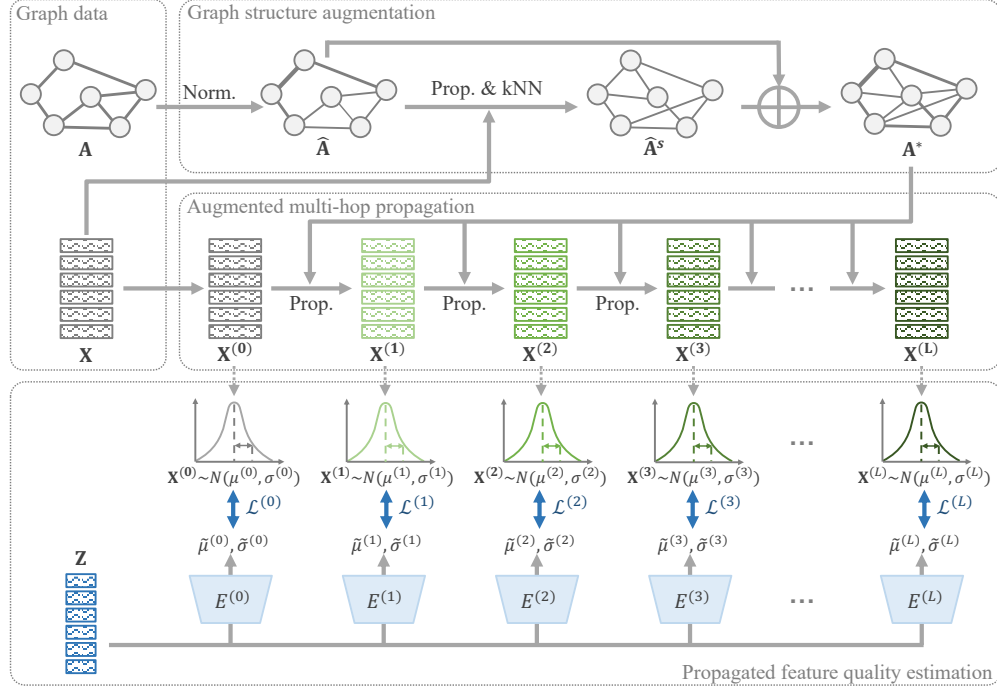
Figure 5: The pipeline of MQE. First, a kNN-based *graph structure augmentation* is conducted to generate the augmented adjacency matrix A^*. Then, the propagated features are acquired by *augmented multi-hop propagation* with A^*. Afterward, in *propagated feature quality estimation*, we take meta representations Z as the condition to estimate the mean μ and standard deviation σ of the distribution of propagated features. Z, finally, serves as the learned representations for downstream tasks.

4 METHODOLOGY

According to the above analyses, the key to addressing the noisy feature problem in UGRL is to fully exploit the denoising mechanism of multi-hop propagation while mitigating the negative impact on representation quality caused by propagation. In line with this objective, we propose a novel UGRL method termed **M**ulti-hop feature **Q**uality **E**stimation (MQE for short) to tackle this challenging problem. Our core idea is to explicitly estimate the quality of propagated features at multiple propagation steps and to learn reliable node representations. The overall pipeline of MQE is illustrated in Fig. 5. Specifically, we first conduct multi-hop feature propagation on an augmented graph structure, which enables us to acquire denoised features from different propagation steps. Then, we introduce Gaussian distribution to model the distribution of propagated features at multiple steps, enabling the estimation of their quality using the standard deviation σ. In MQE, we utilize learnable latent variable Z called "meta representations" to encode the essential information of nodes. Afterward, neural network-based estimators E are employed to approximate the expectation μ and standard deviation σ of the propagated features based on Z. Once MQE is well-trained, the meta representations capture the true feature signals as well as the surrounding structural information of each node, and hence can serve as node representations for downstream tasks. The designs of MQE will be introduced in the following subsections.

4.1 Augmented Multi-Hop Propagation

In Sec. 3, we recognized that long-distance propagation in UGRL can filter out high-frequency noisy information; however, relying solely

on propagation-based GNNs to generate representations may pose a challenge due to the potential propagation of noise. To this end, in MQE, we take the features acquired by multi-hop propagation as the target to be estimated rather than the learned representations.

Specifically, the multi-hop propagated features can be calculated by iteratively. In each iteration, the propagated feature vector of the i-th node v_i can be calculated by

$$\mathbf{x}_i^{(\ell)} = \sum_{v_j \subset \mathcal{N}_i} \hat{a}_{ij} \mathbf{x}_j^{(\ell-1)}, \tag{1}$$

where $\mathbf{x}_i^{(\ell)}$ is the propagated feature of node v_i at the ℓ-th propagation step, $\hat{a}_{ij}$ is the i, j-th entry of the symmetric normalization of the adjacency matrix $\hat{\mathbf{A}}_{ij}$, and $\mathcal{N}_i$ is the neighbor set of v_i. We define $\mathbf{x}_i^{(0)} = \mathbf{x}_i$ and calculate $\mathbf{x}_i^{(1)}, \cdots, \mathbf{x}_i^{(L)}$ via Eq.(1) with a predefined maximum propagation step L.

It is worth noting that in MQE, we employ a non-parameterized propagation scheme without any linear or non-linear feature transformations, which offers several advantages. Firstly, the propagated features can be precomputed during the preprocessing phase, as no learnable parameters are involved. As a result, the computational cost of model training can be acceptable, even if L is large. Furthermore, the non-parameterized design avoids the potential risk of model degradation caused by redundant transformation operations, which have been shown to be detrimental in supervised GNNs [51].

Graph Structure Augmentation. Although propagating features on the original graph structure (defined by $\hat{\mathbf{A}}_{ij}$) is a default selection in the majority of GNNs [17, 44], it may bring several issues in

UGRL scenarios with noisy features. Given the uncertainty often present in real-world graph structures, the original graph inevitably includes noisy edges and misses key connections [3, 15]. Therefore, directly executing propagation on $\hat{\mathbf{A}}_{ij}$ may result in ineffective feature denoising. Moreover, for the nodes that are isolated from other nodes in the original structure, propagation evidently fails to mitigate their noise characteristic issues.

To address the aforementioned issues, in MQE, we utilize graph structure augmentation [6, 39] to enhance the quality of the original structure, and then execute propagation on the augmented graph. Inspired by homophily assumption [19], we extend the neighbor set of each node by including new neighboring nodes with similar features. Specifically, we construct a k-nearest neighbor (kNN) graph $\mathbf{A}^s$ based on the propagated features for graph structure augmentation. Formally, we first add the propagated features at different steps together, and calculate their pair-wise cosine similarity by

$$\mathbf{x}_i^* = \sum_{\ell=0}^{L} \mathbf{x}_i^{(\ell)}, \quad a_{ij}^s = \frac{\mathbf{x}_i^* \cdot \mathbf{x}_j^*}{||\mathbf{x}_i^*|| \cdot ||\mathbf{x}_j^*||}, \quad (2)$$

where a_{ij}^s represents the similarity between the i-th and j-th nodes. Then, to preserve the sparsity of the augmented graph, we select the top-k neighbors with the highest similarity as the extended neighbors of each node. Subsequently, symmetric normalization is applied to the sparsified graph, yielding the adjacency matrix of the supplementary graph, denoted by $\hat{\mathbf{A}}^s$. To harness both the information from the original graph and the supplementary graph collectively, we integrate them to create the augmented graph $\mathbf{A}^*$. Finally, we conduct multi-hop propagation on the augmented graph to obtain more robust propagated features:

$$\mathbf{A}^* = \frac{\hat{\mathbf{A}} + \hat{\mathbf{A}}^s}{2}, \quad \hat{\mathbf{x}}_i^{(\ell)} = \sum_{v_j \subset \mathcal{N}_i^*} a_{ij}^* \hat{\mathbf{x}}_j^{(\ell-1)}, \quad (3)$$

where a_{ij}^* is the i,j-th entry of $\mathbf{A}^*$, $\mathcal{N}_i^* = \{v_j | a_{ij}^* > 0\}$ is the extended node neighbor set, and $\hat{\mathbf{x}}_i^{(\ell)}$ is the propagated features for quality estimation in MQE. Different from the stochastic graph augmentation in other UGRL methods [60, 61], the augmentation in MQE is constant and non-parameterized. Hence, we only need to conduct the augmented multi-hop propagation once during the preprocessing phase.

4.2 Propagated Feature Quality Estimation

With augmented multi-hop propagation, we can obtain propagated features containing the information of different receptive fields. However, as we discussed in Sec. 3.2, the informativeness of the propagated features may vary across different propagation steps, resulting in their uncertain quality. To extract reliable representation from the propagated features at multiple steps, in MQE, we introduce a Gaussian model-based estimation mechanism that aims to explicitly estimate the quality of propagated features.

In the real world, sample features are usually obtained through a complex generative process, which can be approximated as a superposition of Gaussian distributions in large-scale data [4]. Thus, to model the distribution of propagated features, we assume that the features at each step obey a Gaussian distribution, i.e., $\hat{\mathbf{x}}_i^{(\ell)} \sim$

$N(\mu_i^{(\ell)}, (\sigma_i^{(\ell)})^2)$. Under such an assumption, the corresponding probability density function can be written by

$$p\left(\hat{\mathbf{x}}_i^{(\ell)}\right) = \frac{1}{\sqrt{2\pi \left(\sigma_i^{(\ell)}\right)^2}} e^{-\frac{\left(\hat{\mathbf{x}}_i^{(\ell)} - \mu_i^{(\ell)}\right)^2}{2\left(\sigma_i^{(\ell)}\right)^2}}. \quad (4)$$

In Eq. (4), the arithmetic mean $\mu_i^{(\ell)}$ indicates the expectation of true signals within the propagated feature vector $\hat{\mathbf{x}}_i^{(\ell)}$, while the standard deviation $\sigma_i^{(\ell)}$ reflects the quality of $\hat{\mathbf{x}}_i^{(\ell)}$. More concretely, a larger $\sigma_i^{(\ell)}$ indicates a lower quality of $\hat{\mathbf{x}}_i^{(\ell)}$, as it suggests that the features likely experience a greater shift from the true signal.

Given the Gaussian distribution-based modeling for the propagated features, we can transfer the quality estimation task to a variational prediction task for the distribution of $\hat{\mathbf{x}}_i^{(\ell)} \sim N(\mu_i^{(\ell)}, (\sigma_i^{(\ell)})^2)$. Based on the above modeling, our goal is to encode fruitful information of each node from multiple propagation steps into a unified node representation. Thus, for each node v_i we define a learnable vector known as the "meta-representation" $\mathbf{z}_i$ as a consistent condition shared across multiple propagation steps. In the feature quality estimation phase, the meta-representation $\mathbf{z}_i$ can be viewed as a compact latent variable encapsulating node attributes and contextual information.

Specifically, in MQE, $\mathbf{z}_i$ is initialized with random values and then updated concurrently with the model parameters. Given the meta representation $\mathbf{z}_i$ as condition, we establish two estimator networks $E_\mu^{(\ell)}(\cdot)$ and $E_\sigma^{(\ell)}(\cdot)$ to estimate $\mu_i^{(\ell)}$ and $\sigma_i^{(\ell)}$, respectively. In practice, $E_\mu^{(\ell)}(\cdot)$ and $E_\sigma^{(\ell)}(\cdot)$ are both 2-layer multilayer perceptron (MLP) networks. Formally, the conditional likelihood for the ℓ-step propagated features can be written by

$$p\left(\hat{\mathbf{x}}_i^{(\ell)} \mid \mathbf{z}_i\right) = N\left(\tilde{\mu}_i^{(\ell)}, \left(\tilde{\sigma}_i^{(\ell)}\right)^2\right) = \frac{1}{\sqrt{2\pi \left(\tilde{\sigma}_i^{(\ell)}\right)^2}} e^{-\frac{\left(\hat{\mathbf{x}}_i - \tilde{\mu}_i^{(\ell)}\right)^2}{2\left(\tilde{\sigma}_i^{(\ell)}\right)^2}}, \quad (5)$$

where $\tilde{\mu}_i^{(\ell)} = E_\mu^{(\ell)}(\mathbf{z}_i)$ and $\tilde{\sigma}_i^{(\ell)} = E_\sigma^{(\ell)}(\mathbf{z}_i)$ are the estimated mean and standard deviation acquired from the estimators, respectively.

To implement effective feature quality estimation, we optimize the estimators $E = \{E^{(0)}, \cdots, E^{(L)}\} = \{E_\mu^{(0)}, E_\sigma^{(0)}, \cdots, E_\mu^{(L)}, E_\sigma^{(L)}\}$ for every propagation steps as well as the meta representations $\mathbf{Z} = [\mathbf{z}_1^\mathsf{T}, \cdots, \mathbf{z}_n^\mathsf{T}]$ with a variational reconstruction loss. Specifically, we employ a negative logarithmic reconstruction likelihood loss function, which can be written by

$$\begin{aligned}
\mathcal{L} &= \sum_{\ell=1}^{L} \mathcal{L}^{(\ell)} = \sum_{\ell=1}^{L} \sum_{i=1}^{n} -\ln\left(p\left(\mathbf{x}_i^{(\ell)} \mid \mathbf{z}_i\right)\right) \\
&= \sum_{\ell=1}^{L} \sum_{i=1}^{n} \frac{\left(\mathbf{x}_i^{(\ell)} - \tilde{\mu}_i^{(\ell)}\right)^2}{2\left(\tilde{\sigma}_i^{(\ell)}\right)^2} + \ln\left(\tilde{\sigma}_i^{(\ell)}\right),
\end{aligned} \quad (6)$$

where a constant term in the original expression is ignored since it does not affect the optimization of the model. In the loss function, the first term aims to minimize the reconstruction error between

$\tilde{\mu}_i^{(\ell)}$ and the propagated features $\mathbf{x}_i^{(\ell)}$, with a tolerance controlled by $\tilde{\sigma}_i^{(\ell)}$. Concretely, a larger value of $\tilde{\sigma}_i^{(\ell)}$ denotes a more relaxed constraint on the reconstruction error, consequently implying a diminished quality of $\mathbf{x}_i^{(\ell)}$. Our experiments detailed in Sec. 5.3 affirm that $\tilde{\sigma}_i^{(0)}$ effectively serves as an indicator for the noise intensity of features. The second term serves as a regularization component aimed at averting collapse to a trivial solution, where σ becomes excessively large for all propagated features.

Z as Final Representations. During the model training process, $\mathbf{Z}$ progressively and adaptively amalgamates information from various propagated features, ultimately evolving into concise and informative representations for each node. As an accurate estimation of multi-hop propagated features, $\mathbf{Z}$ encompasses not only the true feature signals but also the knowledge from neighboring nodes. Hence, during the inference phase, we can directly utilize $\mathbf{Z}$ as the learned representations by MQE, employing them in various downstream tasks.

Complexity Analysis. The time complexity of MQE primarily consists of two main components: pre-processing and model training. For the pre-processing phase which involves feature propagation and kNN-based augmentation, the complexity is $O\left(Lmd + n^2 d\right)$ ($m = |E|$ is the number of edges) in total, where the first term is for propagation and the second term is for kNN computation. For each training iteration, complexity mainly arises from the two-layer MLP-based estimators. Specifically, the complexity of each $E_\mu^{(\ell)}$ and $E_\sigma^{(\ell)}$ are $O\left(nh(f+d)\right)$ and $O\left(nhf\right)$, respectively, where h and f are the dimensions of hidden layer and meta representation. Therefore, the overall training time complexity is $O\left(Lnh(f+d)\right)$ in total.

5 EXPERIMENTS

In this section, we evaluate the MQE with extensive experiments on different datasets and noisy feature scenarios. Specifically, we aim to answer the following research questions:

RQ1: How effective is MQE in various noisy feature scenarios?
RQ2: Can MQE accurately estimate the noise intensity of nodes?
RQ3: How do pivotal designs impact the performance of MQE?

5.1 Experimental Setup

Datasets. We use node classification as the downstream task to evaluate the informativeness of the learned representations. Our experiments are conducted on 5 benchmark datasets, including 3 citation graph datasets (i.e., Cora, CiteSeer, and PubMed [36]) and 2 co-purchase graph datasets (i.e., Amazon Computers and Amazon Photo [37]). All the datasets are randomly divided into 10%, 10%, and 80% for training, validation, and testing respectively, following the setting in [59, 61].

Evaluation Protocol. For each experiment, we follow the evaluation scheme as in [41, 53, 60]. Specifically, each model is first trained in an unsupervised manner on the full graph with node features. Then, the learned representations are used to train and test an $l2$ regularized logistic regression classifier from SciKit-Learn [34]. We conduct 5 runs for each model and report the mean classification accuracy along with its standard deviation as the metric.

Feature Noise Injection. Since all the datasets do not inherently contain feature noise, we manually injected two types of noise into
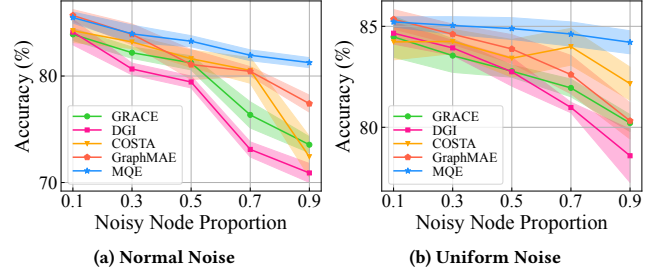

(a) Normal Noise   **(b) Uniform Noise**

Figure 6: Performance comparison in the scenarios with different noisy node fractions α on Cora dataset.

them. Firstly, we randomly select $\alpha \in (0, 1]$ fraction of nodes for perturbation. Then, we sample noise from either standard Gaussian or uniform distribution, and add them to the features of selected points with a coefficient of β (i.e., noise level).

Baselines. We selected representative UGRL methods as the baselines for comparison. Our baselines include 3 types of methods: 1) autoencoder-based UGRL methods, including GAE and VGAE [16]; 2) Contrastive UGRL methods, including DGI [41], GRACE [60], MVGRL [11], and COSTA [53]; 3) Masked graph autoencoder-based UGRL methods, including GraphMAE [13] and MaskGAE [20].

Implementation Details. The experiments are conducted on a machine using Intel(R) Xeon(R) Platinum 8255C CPU@2.50GHz and a single RTX 4000 GPU with 16 GB GPU memory. The operating system of the machine is Ubuntu 20.04. We use Python 3.8, Pytorch 1.11.0, and CUDA 11.3 for the implementation of MQE. Our code is available at https://github.com/Shiy-Li/MQE.

5.2 Performance Comparison (RQ1)

Performance under different noise types and levels. In this experiment, we established two distinct noise settings based on edge sparsity. Specifically, we fix fraction $\alpha = 0.5$, vary $\beta = \{0.3, 0.5, 0.8\}$ (citation network), and vary $\beta = \{1, 5, 10\}$ (co-purchase network) for normal and uniform distribution to simulate different scenarios with noisy features. We also consider the scenario with clean data. On the three larger datasets, we only compare MQE with competitive UGRL methods that have scalability, due to memory limitations. The experimental results are exhibited in Table 1, and from where we have the following observations. 1) MQE generally outperforms the baselines in both clean and noisy feature scenarios. This superior performance demonstrates the effectiveness of multi-hop propagation and quality estimation in addressing the noisy feature issue. 2) Although we model the propagated features with a Gaussian distribution rather than a uniform one, MQE still shows competitive performance in scenarios with uniform noise. This observation underscores the universality of MQE, emphasizing its capability to learn informative representations from graph data with diverse feature noises. 3) As the noise level increases, the performance gap between MQE and the baselines widens, which highlights the capability of MQE in handling extremely noisy data. **Performance under different noisy node fraction α.** To further verify the generalization ability of MQE, we conduct this experiment by varying α from 0.1 to 0.9 while keeping $\beta = 0.5$ constant. From

Table 1: Node classification accuracy. The results of the winner and runner-up are bolded and underlined, respectively.

Dataset	Method	Clean	Normal Noise ($\alpha = 0.5$)			Uniform Noise ($\alpha = 0.5$)		
			$\beta = 0.3$	$\beta = 0.5$	$\beta = 0.8$	$\beta = 0.3$	$\beta = 0.5$	$\beta = 0.8$
Cora	GAE	81.21±0.87	72.24±1.09	70.17±1.50	63.09±1.30	79.28±1.01	77.94±1.13	71.46±2.42
	VGAE	77.05±1.07	70.14±0.87	68.11±0.34	66.53±0.79	76.17±1.15	73.11±0.93	64.47±2.51
	GRACE	85.86±0.63	82.82±0.61	81.02±0.53	79.04±2.54	82.74±0.65	81.24±0.66	79.46±0.62
	DGI	85.38±0.76	81.85±0.79	79.44±0.56	75.28±1.56	84.29±0.29	82.75±0.91	81.07±1.16
	MVGRL	85.63±0.75	83.93±0.71	80.96±0.53	79.13±0.71	85.35±0.65	84.10±0.74	82.54±0.83
	COSTA	85.66±0.56	82.74±0.95	81.62±0.81	79.76±1.09	84.35±0.82	83.42±0.93	81.60±0.88
	GraphMAE	85.61±0.52	82.79±1.21	81.08±1.41	79.84±0.68	84.56±0.62	83.88±0.71	81.81±0.36
	MaskGAE	84.25±0.80	77.48±1.10	72.59±0.31	67.80±0.48	82.65±0.64	81.71±0.89	78.75±0.53
	MQE (Ours)	**85.97±0.42**	**84.71±0.64**	**83.27±0.52**	**82.16±0.55**	**85.44±0.51**	**84.89±0.54**	**83.76±0.28**
CiteSeer	GAE	68.76±0.95	56.09±1.55	53.25±0.68	48.45±0.96	67.36±0.85	63.37±1.97	40.74±7.73
	VGAE	63.70±0.84	48.91±1.64	42.58±2.24	37.34±1.21	62.04±0.86	57.20±2.89	46.60±2.76
	GRACE	73.21±0.71	69.08±0.85	66.94±1.18	64.94±0.63	72.15±0.45	70.32±0.73	65.55±0.72
	DGI	73.39±0.92	65.07±0.67	60.50±1.57	55.82±0.93	72.50±0.78	71.32±0.60	68.10±0.76
	MVGRL	73.19±0.51	67.62±1.35	64.36±0.42	62.14±0.70	72.04±0.58	70.98±0.44	66.88±1.39
	COSTA	72.16±0.59	68.98±0.40	58.08±1.66	53.23±1.48	69.41±0.96	68.78±1.24	62.69±0.70
	GraphMAE	73.82±0.48	70.15±0.24	68.90±0.26	65.90±0.47	70.38±0.61	68.97±0.52	68.50±0.23
	MaskGAE	72.73±0.45	64.68±0.82	58.71±0.88	53.67±0.63	71.86±0.56	69.91±0.93	65.68±0.52
	MQE (Ours)	**74.64±0.53**	**71.53±0.88**	**69.44±0.98**	**66.37±1.27**	**73.79±0.73**	**72.64±1.00**	**72.25±0.58**
PubMed	GRACE	86.10±0.28	72.34±0.52	66.28±0.57	60.31±0.32	81.12±0.38	75.95±0.42	70.11±0.79
	DGI	87.08±0.17	77.19±0.21	71.77±0.44	66.21±0.33	84.49±0.33	81.77±0.30	78.30±0.28
	COSTA	86.98±0.33	75.45±0.45	69.47±0.35	64.86±0.37	83.37±0.34	79.80±0.37	74.99±0.27
	GraphMAE	85.94±0.16	80.49±0.25	77.42±0.24	76.99±0.31	84.23±0.22	82.63±0.37	82.01±0.23
	MaskGAE	87.48±0.20	73.89±0.30	67.81±0.38	62.44±0.57	84.11±0.39	80.30±0.36	75.72±0.58
	MQE (Ours)	**87.73±0.19**	**82.71±0.30**	**81.15±0.23**	**80.37±0.38**	**85.43±0.23**	**84.23±0.22**	**82.52±0.23**

Dataset	Method	Clean	Normal Noise ($\alpha = 0.5$)			Uniform Noise ($\alpha = 0.5$)		
			$\beta = 1$	$\beta = 5$	$\beta = 10$	$\beta = 1$	$\beta = 5$	$\beta = 10$
Photo	GRACE	92.62±0.23	91.65±0.16	88.30±0.36	87.50±0.67	90.96±0.51	87.94±0.41	84.94±0.74
	DGI	93.09±0.08	92.19±0.34	89.19±0.67	88.03±0.28	92.79±0.22	90.99±0.25	89.36±0.12
	COSTA	92.67±0.16	91.18±0.31	86.90±0.64	85.39±0.47	92.41±0.20	90.07±0.57	87.77±0.16
	GraphMAE	93.27±0.31	92.93±0.52	90.18±0.38	88.22±0.76	91.57±0.36	87.59±2.09	82.74±1.13
	MaskGAE	93.42±0.26	91.85±0.25	86.21±0.29	84.15±0.79	93.06±0.24	90.57±0.38	87.80±0.18
	MQE (Ours)	**94.49±0.17**	**93.27±0.17**	**91.19±0.21**	**91.03±0.25**	**93.74±0.31**	**92.01±0.14**	**90.91±0.23**
Computers	GRACE	89.35±0.24	88.21±0.27	84.79±0.41	84.20±0.18	88.75±0.13	83.98±0.14	81.31±0.48
	DGI	89.60±0.22	88.29±0.06	84.24±0.22	82.70±0.42	88.95±0.10	86.53±0.18	83.93±0.43
	COSTA	89.08±0.17	87.43±0.29	82.33±0.19	80.22±0.28	88.44±0.22	86.20±0.31	83.34±0.30
	GraphMAE	90.45±0.16	89.57±0.24	84.90±0.43	84.42±0.21	89.56±0.24	87.17±1.28	83.46±0.46
	MaskGAE	90.01±0.10	87.54±0.28	80.07±0.23	76.57±0.29	89.57±0.08	85.59±0.24	81.96±0.39
	MQE (Ours)	**91.37±0.09**	**89.87±0.17**	**87.72±0.24**	**87.29±0.21**	**90.26±0.24**	**87.74±0.18**	**85.73±0.33**

the results in Fig. 6, we can find the significant performance advantage of MQE when α is large. In contrast, the performance of some baselines (e.g., DGI) sharply declines as more nodes become noisy.

5.3 Quality Estimation Effectiveness (RQ2)

In this subsection, we conduct qualitative experiments to gain insights into the feature quality estimated by MQE. We simulate the noisy feature scenario by setting $\alpha = 0.5$ and $\beta = 0.5$.

Intensity Estimation for Raw Features. In Sec. 4.2, we mention that the estimated standard deviation $\tilde{\sigma}_i^{(0)}$ at the 0-th step propagated features (i.e., the raw features) can be an ideal indicator for noise intensity s_i. To verify the estimation effectiveness, we visualize the correlation between the noisy intensity (defined in Sec. 2.2 and the estimated standard deviation in Fig. 7. Note that we only visualize nodes with $s_i > 0$ and disregard those with clean features.

The figure reveals a strong correlation between the ground-truth noisy intensity and the estimated σ, indicating that MQE can effectively approximate the degree of feature noise for each node. This capability ensures that our method provides an in-depth understanding of the distribution of noisy features, thereby potentially aiding in data engineering and cleaning processes.

Visualization of the Quality of Propagated Features. By examining $\tilde{\sigma}_i^{(l)}$ estimated by MQE at different propagation steps, we can further analyze the impact of propagation on noisy features. As depicted in Fig. 8, the estimated quality for the original features effectively distinguishes between clean nodes and noisy nodes. As the propagation process progresses, the estimated noise intensity of the noise nodes decreases and the noise intensity of some of the clean nodes increases, further confirming our observation and analysis in Sec. 3.
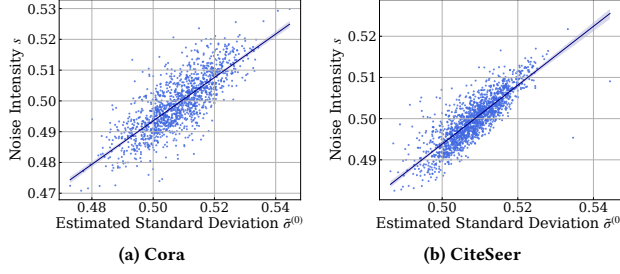
Figure 7: The correlation between estimated standard deviation and feature noise intensity.
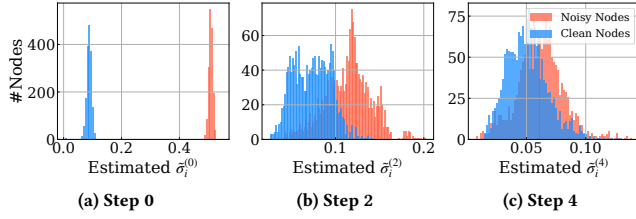

Figure 8: Visualization of the distribution of estimated standard deviation at different propagation steps.

5.4 Ablation Study (RQ3)

To examine the contribution of each component and key design in MQE, we conducted experiments on several variants of MQE, and the results are shown in Table 2. Specifically, "MQE w/o aug" represents the variant without graph structure augmentation; "MQE w/o mh" is the variant that only estimates the quality at the last hop, and "MQE w/o reg" removes the regularization loss term in Eq. (6). We conduct experiments in both clean and noisy ($\alpha = 0.5$, $\beta = 0.8$, normal noise) feature scenarios.

From the results in Table 2, we have the following findings. 1) The performance of "MQE w/o mh" is generally lower than MQE, indicating the significance of considering the propagated information at multiple propagation steps. 2) Without the regularization term, the performance of MQE can drastically degrade, which illustrates that the regularization term is critical in preventing trivial solution issues. 3) In most cases, augmenting the graph structure leads to improvement, particularly in scenarios with noisy features. 4) The performance degradation of the three variants of MQE is generally higher in noisy scenarios than in clean scenarios, further illustrating the importance of these key designs in addressing the noisy feature problem.

6 RELATED WORK

Unsupervised graph representation learning (UGRL) aims to learn node-level representations from graphs by means of label-free cost, and is widely used in various real-world scenarios [7, 29, 43, 56]. Traditional methods such as deepwalk [35], node2vec [9] generate low-dimensional embedding of nodes based on graph topology and random walk strategy. The recent popular contrastive UGRL methods are inspired by contrastive learning and follow the principle of maximizing mutual information to optimally learn node embeddings [11, 41, 60]. Another line of methods termed generative

Table 2: Performance of MQE and its variants.

Dataset	Variant	Clean	Noisy
Cora	MQE w/o aug	84.61±0.33	80.38±0.36
	MQE w/o mh	85.38±0.46	79.53±0.64
	MQE w/o reg	81.96±1.12	30.89±0.51
	MQE	**85.97±0.42**	**82.16±0.55**
CiteSeer	MQE w/o aug	74.32±0.61	63.25±0.71
	MQE w/o mh	73.03±0.65	64.42±0.88
	MQE w/o reg	53.28±4.98	21.54±1.01
	MQE	**74.64±0.53**	**66.37±1.27**
PubMed	MQE w/o aug	**88.00±0.28**	76.48±0.31
	MQE w/o mh	83.93±0.36	76.13±0.31
	MQE w/o reg	85.84±0.35	51.63±0.91
	MQE	87.73±0.19	**80.37±0.38**

UGRL aims to learn node representations by recovering the missing parts of the input data [5, 12, 16, 46]. For example, MGAE [38] and GraphMAE [13] recover hidden structures and attributes by implementing masking strategies on graph structures and node attributes, allowing the model to obtain better node representations [20].

Unlike the aforementioned UGRL models, which are limited to perfect features, MQE focuses on a broader spectrum of real-world scenarios with noisy features. At the methodology level, MQE generates representations through the process of estimation rather than direct propagation via GNNs, which reduces the risk of noise diffusion during propagation.

7 CONCLUSIONS

In this paper, we undertake the first endeavor in unsupervised graph representation learning (URGL) on graph data with noisy features — a challenge that remains largely unexplored in real-world scenarios, despite its inevitability. Through empirical analysis, we uncover the advantages and drawbacks of message propagation in addressing the noisy feature issue, and recognize the importance of estimating the quality of propagated information. Based on these insights, we propose a new UGRL method, termed MQE, which learns reliable node representations by estimating the quality of multi-hop propagated features with a conditional Gaussian model. Extensive experiments have demonstrated the effectiveness of MQE in learning high-quality representations and approximating node-level noise intensity in various noisy feature scenarios.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*. PMLR, 21–29.
[2] Jinyu Cai, Yunhe Zhang, Jicong Fan, and See-Kiong Ng. 2024. LG-FGAD: An Effective Federated Graph Anomaly Detection Framework. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

[3] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems* 33 (2020), 19314–19326.

[4] Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk-Jae Lee. 2019. Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving. In *Proceedings of the IEEE/CVF International conference on computer vision.* 502–511.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[6] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. 2022. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter* 24, 2 (2022), 61–77.

[7] Suyu Ge, Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2020. Graph enhanced representation learning for news recommendation. In *Proceedings of the web conference 2020.* 2863–2869.

[8] Yu Geng, Zongbo Han, Changqing Zhang, and Qinghua Hu. 2021. Uncertainty-aware multi-view representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 7545–7553.

[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD.* 855–864.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[11] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *International conference on machine learning.* PMLR, 4116–4126.

[12] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 16000–16009.

[13] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 594–604.

[14] Xia Hu, Lei Tang, Jiliang Tang, and Huan Liu. 2013. Exploiting social relations for sentiment analysis in microblogging. In *Proceedings of the sixth ACM international conference on Web search and data mining.* 537–546.

[15] 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.

[16] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[17] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR.*

[18] Huan Yee Koh, Anh TN Nguyen, Shirui Pan, Lauren T May, and Geoffrey I Webb. 2023. PSICHIC: physicochemical graph neural network for learning protein-ligand interaction fingerprints from sequence data. *bioRxiv* (2023), 2023–09.

[19] Gueorgi Kossinets and Duncan J Watts. 2009. Origins of homophily in an evolving social network. *American journal of sociology* 115, 2 (2009), 405–450.

[20] Jintang Li, Ruofan Wu, Wangbin Sun, Liang Chen, Sheng Tian, Liang Zhu, Changhua Meng, Zibin Zheng, and Weiqiang Wang. 2023. What's Behind the Mask: Understanding Masked Graph Modeling for Graph Autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 1268–1279.

[21] Rui Li, Shengjie Wang, and Kevin Chen-Chuan Chang. 2012. Multiple location profiling for users and relationships from social network and content. *arXiv preprint arXiv:1208.0288* (2012).

[22] Shiyuan Li, Qingfeng Chen, Zhixian Liu, Shirui Pan, and Shichao Zhang. 2024. Bi-SGTAR: A simple yet efficient model for circRNA-disease association prediction based on known association pair only. *Knowledge-Based Systems* 291 (2024), 111622.

[23] Ke Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, Xinwang Liu, Fuchun Sun, and Kunlun He. 2024. A survey of knowledge graph reasoning on graph types:Static, dynamic, and multi-modal. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[24] Yixin Liu, Thalaiyasingam Ajanthan, Hisham Husain, and Vu Nguyen. 2024. Self-supervision improves diffusion models for tabular data imputation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management.*

[25] Yixin Liu, Kaize Ding, Jianling Wang, Vincent Lee, Huan Liu, and Shirui Pan. 2023. Learning strong graph neural networks with weak information. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 1559–1571.

[26] Yixin Liu, Shiyuan Li, Yu Zheng, Qingfeng Chen, Chengqi Zhang, and Shirui Pan. 2024. ARC: A Generalist Graph Anomaly Detector with In-Context Learning. *arXiv preprint arXiv:2405.16771* (2024).

[27] Yue Liu, Ke Liang, Jun Xia, Xihong Yang, Sihang Zhou, Meng Liu, Xinwang Liu, and Stan Z Li. 2023. Reinforcement graph clustering with unknown cluster number. In *Proceedings of the 31st ACM international conference on multimedia.* 3528–3537.

[28] Yue Liu, Sihang Zhou, Xinwang Liu, Wenxuan Tu, and Xihong Yang. 2022. Improved dual correlation reduction network. *arXiv preprint arXiv:2202.12533* (2022).

[29] Yue Liu, Shihao Zhu, Jun Xia, Yingwei Ma, Jian Ma, Wenliang Zhong, Guannan Zhang, Kejun Zhang, and Xinwang Liu. 2024. End-to-end Learnable Clustering for Intent Learning in Recommendation. *arXiv preprint arXiv:2401.05975* (2024).

[30] Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).

[31] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. 2018. Graph signal processing: Overview, challenges, and applications. *Proc. IEEE* 106, 5 (2018), 808–828.

[32] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407* (2018).

[33] Shirui Pan, Yizhen Zheng, and Yixin Liu. 2024. Integrating graphs with large language models: Methods and prospects. *IEEE Intelligent Systems* 39, 1 (2024), 64–68.

[34] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[35] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD.*

[36] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[37] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[38] Qiaoyu Tan, Ninghao Liu, Xiao Huang, Rui Chen, Soo-Hyun Choi, and Xia Hu. 2022. Mgae: Masked autoencoders for self-supervised learning on graphs. *arXiv preprint arXiv:2201.02534* (2022).

[39] Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. 2023. Federated learning on non-iid graphs via structural knowledge sharing. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 9953–9961.

[40] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[41] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341* (2018).

[42] Yili Wang, Yixin Liu, Xu Shen, Chenyu Li, Kaize Ding, Rui Miao, Ying Wang, Shirui Pan, and Xin Wang. 2024. Unifying Unsupervised Graph-Level Anomaly Detection and Out-of-Distribution Detection: A Benchmark. *arXiv preprint arXiv:2406.15523* (2024).

[43] Yingheng Wang, Yaosen Min, Xin Chen, and Ji Wu. 2021. Multi-view graph contrastive representation learning for drug-drug interaction prediction. In *Proceedings of the Web Conference 2021.* 2921–2933.

[44] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning.* PMLR, 6861–6871.

[45] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[46] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. 2022. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 9653–9663.

[47] Jing-Hua Yang, Chuan Chen, Hong-Ning Dai, Meng Ding, Le-Le Fu, and Zibin Zheng. 2022. Hierarchical representation for multi-view clustering: from intra-sample to intra-view to inter-view. In *Proceedings of The 31st ACM International Conference on Information & Knowledge Management.* 2362–2371.

[48] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* 33 (2020), 5812–5823.

[49] Chunxu Zhang, Ximing Li, Hongbin Pei, Zijian Zhang, Bing Liu, and Bo Yang. 2023. LaenNet: Learning robust GCNs by propagating labels. *Neural Networks* 168 (2023), 652–664.

[50] Lei Zhang, Lele Fu, Tong Wang, Chuan Chen, and Chuanfu Zhang. 2023. Mutual information-driven multi-view clustering. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management.* 3268–3277.

[51] Wentao Zhang, Zeang Sheng, Ziqi Yin, Yuezihan Jiang, Yikuan Xia, Jun Gao, Zhi Yang, and Bin Cui. 2022. Model degradation hinders deep graph neural networks. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining.* 2493–2503.

[52] Wentao Zhang, Ziqi Yin, Zeang Sheng, Yang Li, Wen Ouyang, Xiaosen Li, Yangyu Tao, Zhi Yang, and Bin Cui. 2022. Graph attention multi-layer perceptron. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4560–4570.

[53] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. 2022. COSTA: covariance-preserving feature augmentation for graph contrastive learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2524–2534.

[54] Xin Zheng, Dongjin Song, Qingsong Wen, Bo Du, and Shirui Pan. 2024. Online GNN Evaluation Under Test-time Graph Distribution Shifts. In *The Twelfth International Conference on Learning Representations*.

[55] Xin Zheng, Miao Zhang, Chunyang Chen, Soheila Molaei, Chuan Zhou, and Shirui Pan. 2024. Gnnevaluator: Evaluating gnn performance on unseen graphs without labels. In *Advances in Neural Information Processing Systems*, Vol. 36.

[56] Yizhen Zheng, Shirui Pan, Vincent Lee, Yu Zheng, and Philip S Yu. 2022. Rethinking and scaling up graph contrastive learning: An extremely efficient approach with group discrimination. *Advances in Neural Information Processing Systems* 35

(2022), 10809–10820.

[57] Bingxin Zhou, Yuanhong Jiang, Yuguang Wang, Jingwei Liang, Junbin Gao, Shirui Pan, and Xiaoqun Zhang. 2023. Robust graph representation learning for local corruption recovery. In *Proceedings of the ACM Web Conference 2023*. 438–448.

[58] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems* 33 (2020), 7793–7804.

[59] Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. 2021. An Empirical Study of Graph Contrastive Learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[60] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131* (2020).

[61] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*. 2069–2080.