Multi-Difficulty Measure Curriculum Learning for Heterogeneous Graphs with Noise

Anonymous authors
Paper under double-blind review

Abstract

The use of heterogeneous graphs has gained significant traction for modeling and analyzing complex systems across diverse domains because of their ability to represent various types of entities and relationships. However, these graphs face considerable challenges due to different types of noise, including node feature noise, edge noise, and label noise, which arise from data collection imperfections, inconsistent labeling processes, and graph construction errors. These noises significantly undermine the performance of Graph Neural Networks (GNNs), which rely on high-quality data to learn meaningful patterns. In this paper, we address these challenges by investigating the integration of Curriculum Learning (CL) to enhance the robustness of GNNs against multiple forms of noise in heterogeneous graphs. We propose a novel approach, Multi-Difficulty Measure Curriculum Learning (MDCL), which adaptively incorporates diverse difficulty measures to capture various aspects of heterogeneous graphs, including node features, topological structures, and training dynamics. MDCL utilizes an adaptive weighting mechanism to dynamically balance these difficulty measures, optimizing the learning process in the presence of complex noise. Empirical evaluations on benchmark datasets and GNN architectures demonstrate that MDCL consistently improves the accuracy and robustness of GNNs in scenarios with diverse noise types, establishing it as a promising solution for real-world applications involving heterogeneous graphs.

1 Introduction

Heterogeneous graphs have become an essential tool for representing and analyzing complex systems across various domains due to their ability to capture diverse entities and intricate relationships within a network. Unlike homogeneous graphs, which consist of a single type of node and edge, heterogeneous graphs feature multiple types of nodes and edges, offering a more comprehensive and nuanced representation of real-world systems. This richer structure allows for a deeper understanding of the interactions within a system, making heterogeneous graphs particularly valuable for tasks such as recommendation systems, social network analysis, and biological data modeling Yang et al. (2020).

The diversity and complexity inherent in heterogeneous graphs also introduce significant challenges, particularly related to the presence of various forms of noise. Noise in heterogeneous graphs can manifest in multiple ways, including node feature noise Shi et al. (2022), edge noise Zhang et al. (2024), and label noise Dai et al. (2021); Wei et al. (2023); Wong et al. (2024). Node feature noise arises from inaccuracies in the attributes of graph nodes, such as incomplete or erroneous feature representations. Edge noise, on the other hand, originates from incorrect or missing relationships between nodes, which can distort the graph's topological structure. Label noise, commonly caused by imperfect data collection and inconsistent labeling processes Dai et al. (2021), further exacerbates the problem by altering the supervision required for the training of graph learning models. Addressing these challenges is critical for the advancement of the study and application of heterogeneous graphs.

Traditional Graph Neural Networks (GNNs) for homogeneous graphs, such as GCN Kipf & Welling (2016) and GAT Veličković et al. (2018), are sensitive to these types of noise, which significantly hampers their performance. In heterogeneous graphs, the challenge is even more pronounced due to the added complexity

of multiple node and edge types, making models more susceptible to noise in both structural and feature spaces. Furthermore, GNNs trained on noisy data are highly vulnerable to adversarial attacks, which exploit these weaknesses to degrade performance Bojchevski & Günnemann (2019); Zügner et al. (2018); Jin et al. (2021). Existing GNNs for heterogeneous graphs, such as HAN Wang et al. (2019), RGCN Schlichtkrull et al. (2018), and MAGNN Fu et al. (2020), struggle to effectively handle these diverse types of noise, resulting in reduced accuracy and robustness in learning outcomes. This underscores the need for methods that can address node feature noise, edge noise, and label noise in a unified framework, enabling GNNs to perform reliably in real-world heterogeneous graph settings.

Curriculum learning (CL) Bengio et al. (2009); Zhang et al. (2024); Guo et al. (2018) offers a promising strategy to mitigate the effects of data noise by structuring the training process to begin with simpler, higher-quality samples and gradually incorporating more complex and potentially noisy data Yang et al. (2024); Han et al. (2018); Li et al. (2020); Wei et al. (2020); Liu et al. (2020); Guo et al. (2018). This approach allows models to build a robust foundation before handling more challenging scenarios. In graph learning, CL has been implemented through various models. For example, GNN-CL Li et al. (2024) rely on limited, single-perspective difficulty metrics that inadequately represent the multifaceted nature of noise in heterogeneous graphs, such as node feature noise, edge noise, and label noise. The dependence of these models on labeled information further limits their applicability in contexts with sparse or imbalanced labels, reducing their effectiveness in noisy environments. CLNode Wei et al. (2023) utilizes both local and global difficulty measures but relies on manual adjustment of their proportions through empirical tuning, making it labor-intensive and potentially suboptimal. In contrast, MCCL Vakil & Amiri (2023), while incorporating a broader range of graph difficulty indices and an automated prioritization mechanism, focuses primarily on degree, centrality, and connectivity metrics. However, these metrics fail to capture the heterogeneity inherent in different types of entities and relationships within heterogeneous graphs. Moreover, methods like MTGNN Wu et al. (2020) and DRL Qu et al. (2018) are tailored to specific graph structures, restricting their generalizability and utility in heterogeneous graph settings. Similarly, curriculum learning adaptations for heterogeneous graphs, such as loss-aware training schedules for node classification Wong et al. (2024), often rely on single-difficulty measures and require extensive tuning, limiting their robustness in the presence of multiple types of noise.

To address these challenges, this paper proposes a unified approach that integrates curriculum learning into the training of Graph Neural Networks (GNNs) on heterogeneous graphs, with the goal of mitigating the effects of node feature noise, edge noise, and label noise. Based on the unique characteristics of heterogeneous graphs, we propose a novel method named **M**ulti-**D**ifficulty Measure **C**urriculum **L**earning (**MDCL**), which dynamically addresses these challenges through adaptive learning strategies.

The main contributions of this paper are summarized as follows:

- **Novel Difficulty Measures:** We introduce three new difficulty measures specifically designed to mitigate various types of noise in heterogeneous graphs. These include (1) a feature difficulty measure derived from node embeddings, (2) a topological difficulty measure based on local graph structure (e.g., neighborhoods), and (3) a loss difficulty measure to capture discrepancies during training.

- **Adaptive Multi-Difficulty Measure Weighting Mechanism:** We propose an adaptive weighting mechanism that dynamically adjusts the importance of each difficulty measure based on the specific characteristics of the GNN and dataset. This approach optimizes the learning process in the presence of node feature noise, edge noise, and label noise, enhancing the model's generalization ability in complex scenarios.

- **Comprehensive Experimental Validation:** We conduct extensive experiments on various benchmark datasets and backbone GNNs, and demonstrate that MDCL significantly improves accuracy and robustness, particularly in the presence of diverse types of noise, establishing it as a valuable solution for real-world heterogeneous graph applications.

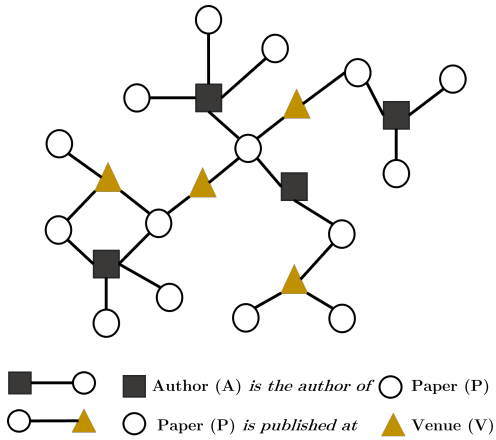


Figure 1: Illustration of one example heterogeneous graph. In this example, the edge type is determined by the types of nodes it connects, providing an implicit specification of the relationships between different node types.

2 Preliminaries

2.1 Heterogeneous Graphs

A heterogeneous graph (Yang et al., 2020) is a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{N}, \mathcal{R}, f_{\mathcal{N}}, f_{\mathcal{R}}\}$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, $\mathcal{N}$ is the set of node types, $\mathcal{R}$ is the set of edge types, $f_{\mathcal{N}} : \mathcal{V} \to \mathcal{N}$ is a mapping from nodes to node types, and $f_{\mathcal{R}} : \mathcal{E} \to \mathcal{R}$ is a mapping from edges to edge types.

When $|\mathcal{N}| = |\mathcal{R}| = 1$, a heterogeneous graph reduces to a homogeneous graph. In this paper, we focus on the case where both $|\mathcal{N}|$ and $|\mathcal{R}|$ are larger than 1. An example is shown in Figure 1, which has 3 types of nodes (author (A), paper (P) and venue (V)), and 4 types of edges specifying the types of nodes connected (A-P/P-A, V-P/P-V).

2.2 Node Classification on Heterogeneous Graphs

The primary task addressed in this paper is node classification on heterogeneous graphs. Node classification involves predicting the labels of nodes based on their features, neighborhood information, and the graph structure. The heterogeneous nature of the graph, with its different node and edge types, adds complexity to this task. Effective models must leverage the heterogeneous information to achieve accurate classification.

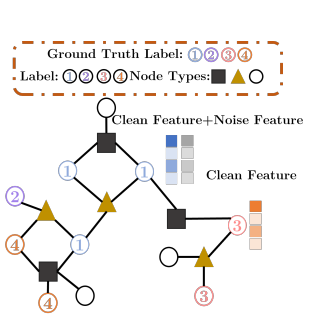
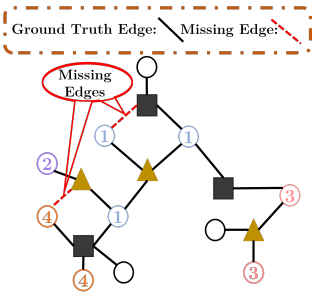
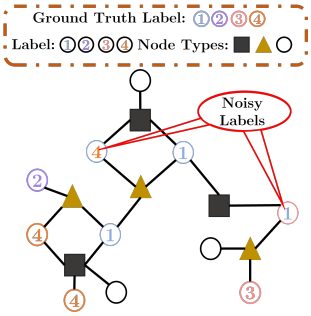
Formally, node classification on a heterogeneous graph is defined as follows. Consider a heterogeneous graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{N}, \mathcal{R}, f_{\mathcal{N}}, f_{\mathcal{R}}\}$. Suppose its node types are given as $\mathcal{N} = \{n_0, \ldots, n_{|\mathcal{N}|-1}\}$, where $|\mathcal{N}|$ is the size of $\mathcal{N}$, and each $\mathcal{V}_i = \{v \in \mathcal{V} : f_{\mathcal{N}}(v) = n_i\}$ denotes the set of all nodes with node type n_i. Suppose each type of nodes n_i is assigned a label space $\mathcal{L}_i$, node classification on heterogeneous graphs (Wang et al., 2019) aims to learn a labelling function $l_i : \mathcal{V}_i \to \mathcal{L}_i$ that maps each node in $\mathcal{V}_i$ to a label in $\mathcal{L}_i$.

For simplicity, we focus on the task of classifying a single type of nodes, denoted as n_0, which we call target nodes. The node classification problem is thus reduced to learning the labeling function for these target nodes.

2.3 Noises in Heterogeneous Graphs

A significant challenge in heterogeneous graph node classification is the presence of noise in the data, which can degrade model performance. This noise can take various forms and affect different aspects of the graph, including node features, labels, and edges. These noise sources can originate from different distributions, which complicate the task of learning accurate node classifications. To capture these issues, we represent

Table 1: Illustration of different types of noise and their corresponding difficulty measures.

Noise Target	Node Feature	Edge	Label
Definition	A proportion of node features are perturbed with noise.	A proportion of edges can be unknown (i.e., missing) in the heterogeneous graph.	A proportion of labels can be corrupted as closely related labels or even unrelated labels.
Illustration			
Difficulty Measure	D_{feat} (Section 3.1)	D_{topo} (Section 3.2)	D_{loss} (Section 3.3)

a noisy and incomplete heterogeneous graph as $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}}, \tilde{\mathcal{N}}, \tilde{\mathcal{R}}, \tilde{f}_{\mathcal{N}}, \tilde{f}_{\mathcal{R}})$, where $\tilde{\mathcal{V}}$ is the set of nodes with noisy features $\tilde{\mathbf{X}}$, $\tilde{\mathcal{E}} \subseteq \mathcal{E}$ is the observed set of edges with some missing connections, and $\mathcal{M}_e$ denotes the set of missing edges. The node types $\tilde{\mathcal{N}}$ are a noisy version of $\mathcal{N}$, representing inaccuracies in node type assignments, while $\tilde{\mathcal{R}}$ is the noisy set of edge types, indicating errors or ambiguities in edge type definitions. Similarly, $\tilde{f}_{\mathcal{N}} : \tilde{\mathcal{V}} \to \tilde{\mathcal{N}}$ is a noisy mapping from nodes to node types, and $\tilde{f}_{\mathcal{R}} : \tilde{\mathcal{E}} \to \tilde{\mathcal{R}}$ is a noisy mapping from edges to edge types. The presence of these issues necessitates the development of robust models that can handle noisy and incomplete data while maintaining high performance in node classification tasks. Table 1 provides an intuitive illustration of the various noise types and the corresponding difficulty measures that we propose to address these challenges.

3 Proposed Method

In this section, we introduce our MDCL (Figure 2), designed to implement curriculum learning on heterogeneous graphs by incorporating various difficulty measures. The key idea of curriculum learning is to reduce the impact of noise by identifying node difficulties, allowing the model to learn from easy and correct samples first. An intuitive way to define node difficulty is to rely on the current performance of the classification model: nodes that are correctly classified are labeled as "easy," whereas those that are incorrectly classified are labeled as "difficult." Building on this concept, we propose three categories of difficulty measures : (i) those computed from node embeddings that capture the semantic and structural features of nodes (Section 3.1), (ii) those computed from topological structures that leverage graph connectivity and structural roles (Section 3.2), and (iii) those computed from label loss that quantify classification errors to directly assess difficulty (Section 3.3). These difficulty measures are designed to address and mitigate issues related to label noise, node feature noise, and edge noise respectively. Then, Section 3.4 introduces an adaptive weighting mechanism that integrates these difficulty measures into a unified comprehensive difficulty score by taking a weighted average of the individual difficulty measures. This overall difficulty score is used to guide the sequential sampling of nodes, enabling a progression from easier to more challenging instances. The complete algorithm is detailed in Section 3.5.

3.1 Measuring Difficulty from Node Embeddings

To effectively measure the difficulty of a target node in heterogeneous graphs, we propose an "augmented" representation h'_u for each target node u that incorporates information from its neighbors with different node types. This approach is specifically designed to address the challenge of noisy node features in graphs by

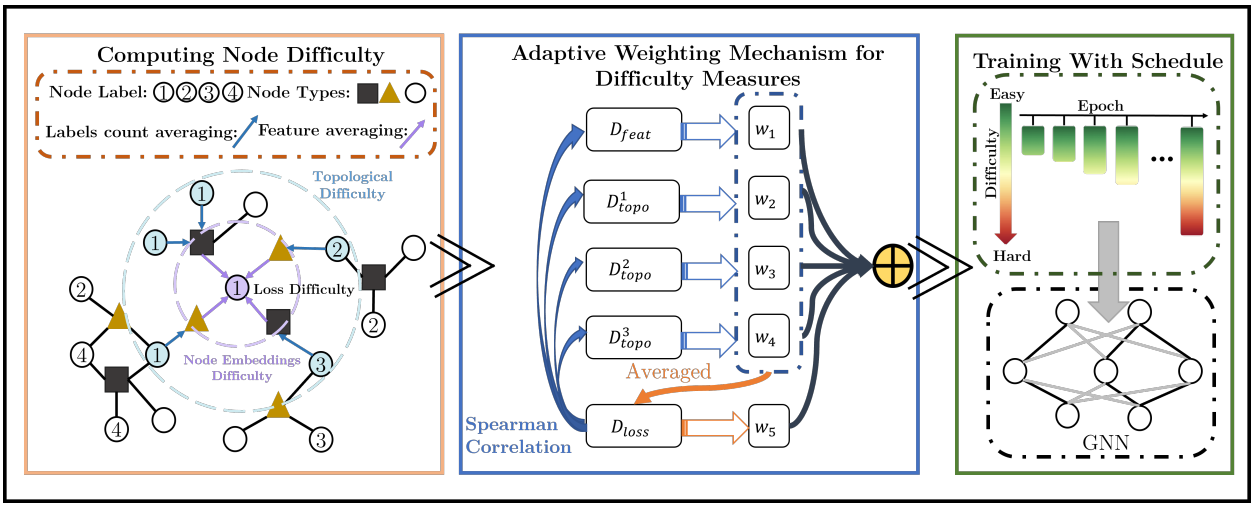


Figure 2: The overview of MDCL. We assess the difficulty of target nodes using a range of difficulty measures: feature difficulty (D_{feat}), topological difficulty ($D^1_{topo}, D^2_{topo}$and D^3_{topo}) and loss difficulty (D_{loss}), corresponding to three different node types. Each measure is adaptively weighted by analyzing its correlation with loss difficulty.

leveraging the diverse and complementary information provided by neighbors of various types. In heterogeneous graphs, different node types contribute uniquely to a node's representation, and our method captures this complexity to better assess the difficulty of nodes with noisy or incomplete features. Specifically, we define the augmented representation as follows:

$$h'_u = h_u + \sum_{v:(u,v)\in\mathcal{E}, f_{\mathcal{N}}(v)\neq n_0} h_v.$$

For each class c, we similarly compute an averaged augmented representation h'_c for all nodes in class c:

$$h'_c = \textsc{Avg}(h'_v \mid Y[v] = c), \tag{1}$$

where $\textsc{Avg}(\cdot)$ denotes a function that returns the average of the input representations.

Using h'_u and h'_c, we define the following difficulty measure:

$$D_{feat}(u) = 1 - \frac{\exp(h'_u \cdot h'_{c_u})}{\max_{c\in\mathcal{C}} \exp(h'_u \cdot h'_c)}. \tag{2}$$

In other words, we compare how similar the augmented representation h'_u of a node u is to the average representation h'_{c_u} of its ground-truth label c_u among all classes $c \in \mathcal{C}$. Our approach extends beyond existing methods, such as CLNode Wei et al. (2023), which measures node difficulty by computing the similarity between a target node's representation h_u and the average representation h_c of all nodes in class c, i.e., $h_c = \textsc{Avg}(h_v \mid Y[v] = c)$. This can be interpreted as utilizing a prototypical classifier Snell et al. (2017) on node embeddings. While this measure can be extended to heterogeneous graphs, it fails to consider the features of other node types that may contribute to more accurate predictions, which is addressed by our augmented representation.

3.2 Measuring Difficulty from Topological Structures

While node features naturally encode important information regarding node difficulty, a critical shortcoming of relying solely on these difficulty measures is the neglect of topological structures in heterogeneous graphs. To address this gap and specifically tackle the challenge of edge noise (i.e., missing edges), we introduce a novel difficulty measure for nodes based on their neighborhood topology.

In homogeneous graphs, the concept of a neighborhood is straightforward. However, in heterogeneous graphs, it becomes more complex due to the presence of different node types and the possibility that direct edges between target nodes may not exist or may be noisy. To account for these challenges, we extend the definition of a neighborhood to incorporate various types of nodes and their interactions within the heterogeneous graph. By doing so, our method not only captures the structural complexity but also mitigates the impact of edge-related noise, providing a more robust evaluation of node difficulty.

Let the type of target nodes be n_0, and let the other types of nodes in the heterogeneous graph be $n_1, n_2, \ldots, n_{|\mathcal{N}|-1} \in \mathcal{N}$, where $|\mathcal{N}|$ is the total number of distinct node types. If there exist edges between two nodes of type n_0, we define the neighborhood of a node u (where $f_{\mathcal{N}}(u) = n_0$) as:

$$\mathcal{N}_0(u) = \{v \mid v = u \text{ or } (u, v) \in \mathcal{E}\}.$$

In addition to nodes of type n_0, node u may also connect to nodes of other types. Consider any other node type n_i that u is connected to. In this case, the edge type $(n_0, n_i) \in \mathcal{R}$, and we use nodes of type n_i as intermediaries to link u to other nodes of type n_0. Thus, we define the neighborhood of u with respect to node type n_i as:

$$\mathcal{N}_i(u) = \{v \mid v = u \text{ or } \exists x \in \mathcal{V}, \text{ s.t. } f_{\mathcal{N}}(x) = n_i, (u, x), (v, x) \in \mathcal{E}\}.$$

Using these definitions, for each training node u, we calculate its difficulty $D^i_{topo}(u)$ based on the label distribution within its neighborhood defined by edges to nodes of type n_i as follows:

$$P_{c,i}(u) = \frac{|\{Y[v] = c \,|\, v \in \hat{\mathcal{N}}_i(u)\}|}{|\hat{\mathcal{N}}_i(u)|}, \tag{3}$$

$$D^i_{topo}(u) = -\sum_{c \in C} P_{c,i}(u) \log(P_{c,i}(u)). \tag{4}$$

Here, $P_{c,i}(u)$ represents the proportion of nodes in the neighborhood $\hat{\mathcal{N}}_i(u)$ that belong to class c. The difficulty measure D^i_{topo} is computed using Shannon entropy. Nodes with more diverse neighbors are naturally more challenging to classify, as they tend to belong to more complex regions of the graph. Consequently, a larger D^i_{topo} value indicates a more diverse neighborhood, making the node more challenging for the model to learn.

Since we obtain multiple D^i_{topo} values corresponding to different node types, a straightforward way to aggregate all these difficulty measures is to take their average. For any node u, we define $D_{topo}(u)$ as:

$$D_{topo}(u) = \frac{1}{|\mathcal{N}|} \sum_{i=0}^{|\mathcal{N}|-1} D^i_{topo}(u). \tag{5}$$

This aggregated measure provides a comprehensive view of the node's topological difficulty, accounting for all possible node types and their interactions within the heterogeneous graph.

3.3 Measuring Difficulty from Node Loss

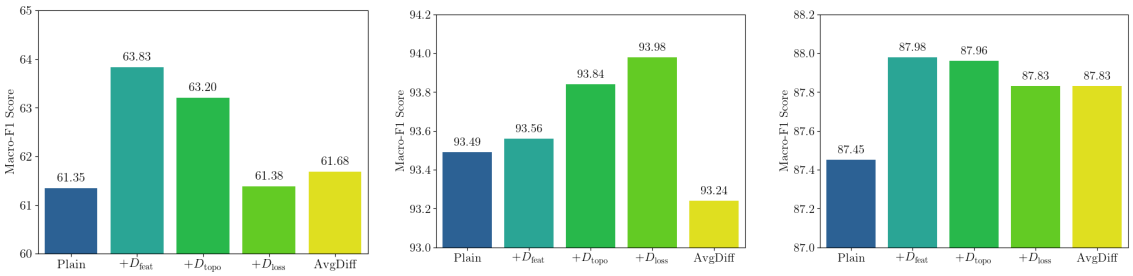
To address the challenge of noisy labels in graphs, we propose evaluating node difficulty from the perspective of training dynamics. Specifically, we leverage a loss-based difficulty measure $D_{loss}(u)$, which is computed using the node embedding h_{u_i} and is commonly employed in existing curriculum learning methods Han et al. (2018):

$$D_{loss}(u) = \frac{1}{E} \sum_{i=1}^{E} \mathcal{L}(Y[v], f(h_{u_i})), \tag{6}$$

where $\mathcal{L}$ represents the loss function, $Y[v]$ is the ground truth label, and $f(h_{u_i})$ is the model's predicted output.

This measure directly relates to the impact of a node's label on the model's training process: a higher $D_{loss}(u)$ indicates that the node contributes significantly to the overall training loss, often signaling instances where the model struggles to align its predictions with potentially noisy or ambiguous labels. By capturing this relationship, $D_{loss}(u)$ serves as a meaningful indicator of the difficulty arising from noisy labels, helping to guide the model's focus during training.

3.4 Adaptive Weighting Mechanism for Difficulty Measures



(a) (HAN, IMDB). (b) (GAT, DBLP). (c) (GAT, ACM).

Figure 3: Comparison of backbone GNN performance on heterogeneous graphs using CL methods with single-difficulty measure and a CL method with uniform-weighted multi-difficulty measure across different datasets. The subfigures show the results for each combination of (backbone GNN, dataset).

When dealing with multiple difficulty measures, a key challenge is determining which measure will perform best for a given scenario. Figure 3 compares the performance of these difficulty measures across different datasets and model architectures. We consider the following methods: (i) **plain**, which does not employ any curriculum learning, (ii) $+\mathbf{D_{feat}}$, which uses D_{feat} defined in Eq. equation 2, (iii) $+\mathbf{D_{topo}}$, which uses D_{topo} defined in Eq. equation 5 and (iv) $+\mathbf{D_{loss}}$, which uses D_{loss} defined in Eq. equation 6 as the difficulty measure. In addition to these individual difficulty measures, we consider a straightforward baseline that averages all these difficulty measures, referred to as **Uniform**. However, while at least one curriculum learning method demonstrates improvement over the baseline in each setting, none consistently outperforms it across all scenarios. In fact, simply averaging the difficulty measures can sometimes perform worse than the plain baseline, such as when using the GAT model on the IMDB dataset.

Motivated by the limitations of using a single-difficulty measure or a uniform averaging approach, we propose an adaptive weighting mechanism to dynamically assign weights to each difficulty measure, enhancing the overall performance of curriculum learning on heterogeneous graphs.

Let the different difficulty measures be denoted as $d_1, \ldots, d_n$. To compute the weighted difficulty measure d, we define it as:

$$d = \sum_{i=1}^{n} w_i d_i, \tag{7}$$

where w_i is the weight assigned to each difficulty measure d_i, which is determined by calculating the relationship between each difficulty measure and the loss difficulty D_{loss}. For each difficulty measure d_i (where $d_i \neq D_{loss}$), we first calculate Spearman's rank correlation with D_{loss}. The weight w_i is then set to the correlation value $\text{cor}(d_i, D_{loss})$. For the loss difficulty measure D_{loss}, the weight w_{loss} is calculated as the average of all other correlation scores:

$$w_i = \begin{cases} \text{cor}(d_i, D_{loss}), & \text{if } d_i \neq D_{loss}, \\ \frac{1}{n-1} \sum_{\substack{j=1 \\ j \neq \text{loss}}}^{n} \text{cor}(d_j, D_{loss}), & \text{if } d_i = D_{loss}. \end{cases} \tag{8}$$

This adaptive weighting mechanism allows the curriculum learning process to effectively prioritize the most relevant difficulty measures based on their relationship with the training dynamics, thereby enhancing the model's robustness and performance across diverse settings.

3.5 Complete Algorithm

Nodes with higher noise levels are typically assigned greater difficulty, as noisy nodes are harder for the model to classify correctly. Based on the calculated difficulty of each node, we implement a curriculum-based training strategy to enhance the performance of the GNN model, as illustrated in Figure 2. To distinguish this model from the original model f_1, we refer to the curriculum-trained model as f_2. The curriculum training then progresses from easy to difficult nodes. Specifically, we first sort the training set $\mathcal{V}_L$ in ascending order based on node difficulty, and introduce a pacing function $g(t)$ to determine the proportion of nodes to be selected for training at each epoch. Let λ_0 represent the initial fraction of the easiest nodes included, and T be the epoch when $g(t)$ first reaches 1. The pacing function $g(t)$ can then take the following form:

$$g(t) = \min(1, \lambda_0 + (1 - \lambda_0) * \frac{t}{T}). \tag{9}$$

After reaching $t = T$, the training does not stop immediately. Instead, it continues to ensure that the GNN model f_2 fully assimilates the knowledge of all nodes in the training set $\mathcal{V}_L$.

The complete training process is summarized in Algorithm 1. Our method generally involves two rounds of training. In the first round, the model is trained like a standard GNN on heterogeneous graphs, without any curriculum learning. Once the initial model is trained, we use it to compute the different difficulty measures for all training nodes. These difficulty measures are then used in the second round to implement the curriculum learning strategy, enabling the model to progressively learn from easier to more challenging nodes, thereby improving robustness and generalization.

Algorithm 1 The complete algorithm of MDCL.

Input: A heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, the labeled node set $\mathcal{V}_L$, the input labels Y_L, the backbone GNN model, the hyper-parameters α, λ_0, T.
Output: The predicted labels $\hat{Y}$.
Initialize parameters of two GNN models f_1 and f_2;
Train f_1 on $(\mathcal{G}, \mathcal{V}_L, Y_L)$;
for $u \in \mathcal{V}_L$ **do**
    Calculate:
    Feature difficulty $D_{feat}(u) \leftarrow$ Eq.equation 2;
    Topological difficulty $D^i_{topo}(u) \leftarrow$ Eq.equation 4;
    Loss difficulty $D_{loss}(u) \leftarrow$ Eq.equation 6;
end for
Compute weights of each difficulty measure $w_i \leftarrow$ Eq.equation 8;
Compute node difficulty by $d(u) = \sum_i w_i d_i(u)$;
Sort $\mathcal{V}_L$ according to node difficulty in ascending order;
Let $t = 1$;
while $t < T$ or not converge **do**
    $\lambda_t \leftarrow g(t)$;
    Generate training subset $\mathcal{V}_t \leftarrow \mathcal{V}_L[1, ..., \lfloor \lambda_t \cdot l \rfloor]$;
    Use f_2 to predict the labels Y_t;
    Calculate loss $\mathcal{L}$ on $\{Y_t[v], Y_L[v] \,|\, v \in \mathcal{V}_t\}$;
    Back-propagation on f_2 to minimize $\mathcal{L}$;
    $t \leftarrow t + 1$;
end while
Predict $\hat{Y}$ with f_2;

4 Experiments

In this section, we conduct experiments under different settings to evaluate the performance of the proposed method MDCL[1].

[1]The codes of MDCL are provided at `https://anonymous.4open.science/r/MDCL-2376`.

4.1 Experimental Setup

Datasets. We conduct experiments on three heterogeneous graph benchmark datasets: **IMDB**, **ACM**, and **DBLP**, which are frequently used in studies on heterogeneous graphs Wang et al. (2019); Hu et al. (2020); Fu et al. (2020). We follow the data loading and splitting protocols outlined in Lv et al. (2021).

Table 2: Statistics of datasets used in this paper.

Datasets	Classes	# Nodes	# Node Types	# Edges	# Edge Types
IMDB	5	21,420	4	86,642	6
ACM	4	10,942	4	547,872	8
DBLP	3	26,128	4	239,566	6

Baselines. We compare our MDCL with the following curriculum learning methods: (i) **Plain**, which does not employ any curriculum learning; (ii) **Loss** Bengio et al. (2009); Wong et al. (2024), which uses D_{loss} defined in Eq. equation 6 as the difficulty measure; (iii) **CLNode** Wei et al. (2023); and (iv) **MCCL** Vakil & Amiri (2023). CLNode and MCCL are originally designed for homogeneous graphs, and when applying them to heterogeneous graphs, we simplify the graph structure by discarding all edge types, effectively making it homogeneous.

Backbone GNNs. We apply the above-mentioned curriculum learning methods to four backbone GNNs: Graph Convolutional Network (**GCN**), Graph Attention Network (**GAT**), Relational GCN (**RGCN**), and Heterogeneous Attention Network (**HAN**). GCN and RGCN are convolution-based models, while GAT and HAN incorporate attention mechanisms to enhance information aggregation. HAN particularly leverages metapath structures in heterogeneous graphs in its architecture.

Noises Setting. We consider various types of noise in graph-structured data to evaluate model robustness under different levels of perturbation. Noise is applied to a proportion η of node features, edges and labels, with each type governed by a well-defined probabilistic model, as described below:

- **Node feature noise** is introduced by perturbing the feature vectors of selected nodes. The noise is sampled from either a standard normal distribution $\mathcal{N}(0, 1)$ (**Gaussian**) or a uniform distribution $\mathcal{U}(-1, 1)$ (**Uniform**) and added to the node's feature vector $\mathbf{x}_i$.
- **Edge noise** is introduced by randomly deleting a proportion η of edges. For each directed edge $(u, v) \in \mathcal{E}$, there is a probability η that the edge is missing (**Missing**).
- **Label noise** is introduced by corrupting the labels of selected nodes using two strategies: **Uniform** where a node's label is randomly replaced with a label from the label set $\mathcal{Y}$, ensuring the new label differs from the original; and **Pairwise** where a node's label is flipped according to a predefined pairwise mapping $\mathcal{M}$, simulating systematic mislabeling between related classes.

We also consider a mixed-noise setting, where all types of noise are combined, simulating real-world scenarios where the type and intensity of noise are often unknown and unpredictable. In this setting, the noise rate η remains consistent across all types to ensure a uniform level of perturbation.

Evaluation Metrics. We report macro-F1 as the evaluation metric. Macro-F1 equally considers all classes, emphasizing performance on both majority and minority classes, making it suitable for imbalanced datasets.

4.2 Performance Comparison

Table 3: Performance (macro-F1) comparison of models under 25% noise rate on the IMDB dataset.

Noise Target		Node Feature		Label		Edge	Mixed
Noise Type		Gaussian	Uniform	Pairwise	Uniform	Missing	
GAT	Plain	62.20	62.45	61.60	62.43	63.41	60.36
	Loss	62.09	62.51	60.65	60.78	63.07	60.86
	CLNode	62.19	62.30	**62.72**	62.86	63.40	61.28
	MCCL	61.63	61.53	59.37	60.79	62.88	60.50
	MDCL	**62.78**	**62.86**	62.20	**63.11**	**63.42**	**61.81**
GCN	Plain	61.47	60.16	60.34	63.11	62.01	61.62
	Loss	61.76	**60.75**	58.43	61.35	62.48	61.83
	CLNode	62.17	60.22	61.32	64.15	**63.27**	62.00
	MCCL	61.96	59.95	57.92	60.60	60.63	61.83
	MDCL	**62.28**	60.50	**61.42**	**64.30**	63.20	**62.42**
HAN	Plain	57.64	60.00	62.01	62.86	62.77	57.03
	Loss	57.24	59.87	61.78	**63.00**	62.46	57.69
	CLNode	57.70	60.26	61.90	62.55	62.09	57.42
	MCCL	57.50	59.62	**62.46**	58.45	62.46	57.69
	MDCL	**57.72**	**60.48**	62.28	62.97	**63.81**	**57.81**
RGCN	Plain	55.35	56.67	58.65	59.36	62.97	54.77
	Loss	56.58	58.15	**60.05**	62.17	63.15	55.66
	CLNode	55.96	57.67	59.05	62.92	62.50	56.13
	MCCL	57.75	**58.42**	58.94	**64.01**	63.02	56.83
	MDCL	**58.18**	58.35	59.79	63.25	**63.21**	**56.95**

Performance Across Noise Types. We evaluate the performance of various methods under different noise conditions, including node feature noise, label noise, edge noise, and mixed-noise scenarios. We provide a comprehensive comparison across multiple datasets, including IMDB (Table 3), DBLP (Table 4), and ACM (Table 5), as well as various GNN architectures such as GAT, GCN, HAN, and RGCN. The results underscore the robustness and effectiveness of our proposed MDCL framework compared to other baselines under these challenging conditions. Although no single baseline demonstrates consistent state-of-the-art performance across all noise types and datasets, our method, MDCL, achieves an average improvement of 1.15% over the plain baseline across individual noise types. In real-world scenarios, the type of noise present in graph data is often unknown and typically involves a mixture of noise types. Even under such challenging conditions, MDCL exhibits superior performance in the mixed-noise scenario, achieving a 1.92% improvement over the plain baseline and outperforming the second-best baseline (i.e., Loss) by 0.84%. This result highlights the capability of MDCL to adapt to and effectively handle complex noise environments, further establishing its utility in practical applications.

Performance Across Noise Rates. We also explore how varying noise rates affect model performance. Figure 4 depicts the macro-F1 scores of RGCN model on the ACM dataset under different noise rates. We can first see that MDCL achieves the best performance from the start without any additional noise, and maintains a top-ranking position as the noise level increases, demonstrating its robustness in handling different kinds of noise.

4.3 Ablation Studies

In this section, we empirically analyze the contribution of different components proposed in our method.

Contribution of Difficulty Measures. To explore the impact of various difficulty measures, we conducted experiments using RGCN as the backbone model on the IMDB dataset under different noise condi-

Table 4: Performance (macro-F1) comparison of models under 25% noise rate on the DBLP dataset.

Noise Target		Node Feature		Label		Edge	Mixed
Noise Type		Gaussian	Uniform	Pairwise	Uniform	Missing	
GAT	Plain	85.74	88.80	89.15	89.82	89.61	84.30
	Loss	85.74	88.87	**89.40**	89.75	**89.86**	84.61
	CLNode	**85.92**	88.88	88.77	89.86	89.68	84.37
	MCCL	83.24	87.85	87.85	88.80	89.61	84.27
	MDCL	85.77	**89.08**	89.26	**89.89**	89.72	**84.65**
GCN	Plain	87.68	87.46	86.94	87.01	87.50	85.53
	Loss	87.89	87.64	85.81	86.97	88.13	85.92
	CLNode	87.78	87.78	86.34	87.46	87.46	85.49
	MCCL	86.73	87.18	86.44	86.62	87.50	85.63
	MDCL	**87.92**	**87.82**	**87.29**	87.32	**88.27**	**85.95**
HAN	Plain	80.18	81.30	77.08	77.56	83.38	76.09
	Loss	80.30	82.04	76.73	77.66	83.63	78.31
	CLNode	75.73	81.90	77.11	77.77	82.71	78.06
	MCCL	80.35	70.95	76.65	67.10	82.85	77.22
	MDCL	**80.51**	81.62	**77.18**	**77.94**	**83.66**	**78.35**
RGCN	Plain	86.84	90.96	81.09	86.16	91.68	84.70
	Loss	88.04	91.43	85.80	87.32	92.53	84.94
	CLNode	88.35	**92.06**	86.01	87.50	92.56	85.65
	MCCL	89.05	91.62	**86.29**	86.57	92.71	84.82
	MDCL	**89.55**	91.80	86.06	**87.63**	**92.77**	**85.77**

tions. The results are presented in Table 6. For instance, in noisy feature scenarios, the $+D_{feat}$ method outperformed the plain baseline. Similarly, under label noise, the $+D_{loss}$ method demonstrated superior performance, highlighting its robustness to such conditions. These findings illustrate that each measure is particularly effective in specific scenarios and can enhance the model's learning through curriculum learning.

Contribution of Adaptive Weighting Mechanism. To further understand the impact of the adaptive weighting mechanism in MDCL, we compare it with: (i) **Plain**, which trains backbone GNNs without curriculum learning; (ii) $+\mathbf{D_{feat}}$, $+\mathbf{D_{topo}}$, and $+\mathbf{D_{loss}}$, which train backbone GNNs using curriculum learning with a single-difficulty setting corresponding to D_{topo}, D_{feat}, and D_{loss}, respectively; and (iii) **AvgDiff**, which trains backbone GNNs with curriculum learning using a uniform mechanism that averages D_{feat}, D_{loss}, and D_{topo}. Table 6 presents the results obtained across various noise conditions. The results demonstrate the superior consistency of MDCL's adaptive weighting mechanism compared to uniform weighting. Notably, in some cases, it can even leverage other difficulty measures to outperform methods specifically designed for certain scenarios. For instance, under the condition of Gaussian noise added to node features, MDCL surpasses $+\mathbf{D_{feat}}$ by 0.44%. It is worth noting that in real-world scenarios, noise often comprises a mixture of multiple types. As shown in the "All" category of the table, MDCL achieves the best performance in such cases, underscoring the importance and effectiveness of the adaptive weighting mechanism.

4.4 Visualization of Difficulty Measure Weights

To better understand how the adaptive mechanism enhances accuracy, we analyze how it assigns weights across different difficulty measures, adapting to changes in the backbone GNNs or datasets. We visualize the learned weights to explore this behavior.

Figure 5 illustrates the weights assigned to various difficulty measures across different datasets when training the RGCN model. The results demonstrate that $+D_{feat}$ plays a more significant role in the IMDB dataset, reflecting the importance of feature-based information in this domain. In contrast, $+D_{topo}$ and $+D_{loss}$ exhibit

Table 5: Performance (macro-F1) comparison of models under 25% noise rate on the ACM dataset.

Noise Target		Node Feature		Label		Edge	Mixed
Noise Type		Gaussian	Uniform	Pairwise	Uniform	Missing	
GAT	Plain	90.79	91.97	90.13	88.57	91.98	87.91
	Loss	**92.02**	92.21	89.75	89.24	92.05	90.98
	CLNode	90.08	92.49	90.27	89.66	92.06	91.93
	MCCL	90.18	92.48	89.57	89.90	91.93	91.03
	MDCL	91.88	**92.68**	**90.75**	**90.23**	**92.06**	**92.03**
GCN	Plain	92.30	92.26	90.93	92.35	92.16	91.21
	Loss	92.26	91.97	90.84	91.97	92.07	91.74
	CLNode	92.59	92.49	90.46	91.78	92.12	91.60
	MCCL	92.49	92.07	90.27	91.97	92.12	91.64
	MDCL	**92.87**	**92.68**	**91.23**	**92.45**	**92.45**	**91.97**
HAN	Plain	83.32	**81.06**	80.01	81.33	84.89	76.10
	Loss	83.71	80.64	80.48	81.10	80.55	76.25
	CLNode	72.29	80.73	79.40	81.47	84.99	76.20
	MCCL	79.84	78.71	77.37	77.32	83.47	76.30
	MDCL	**84.99**	**81.12**	**80.86**	**82.32**	**85.36**	**77.71**
RGCN	Plain	84.44	87.03	82.45	85.19	84.19	82.77
	Loss	84.45	**87.37**	82.99	84.61	**85.81**	81.67
	CLNode	84.43	86.71	**84.06**	85.30	85.21	82.77
	MCCL	84.07	86.75	81.27	83.96	85.29	84.10
	MDCL	**85.40**	87.07	83.20	**85.81**	84.50	**84.27**

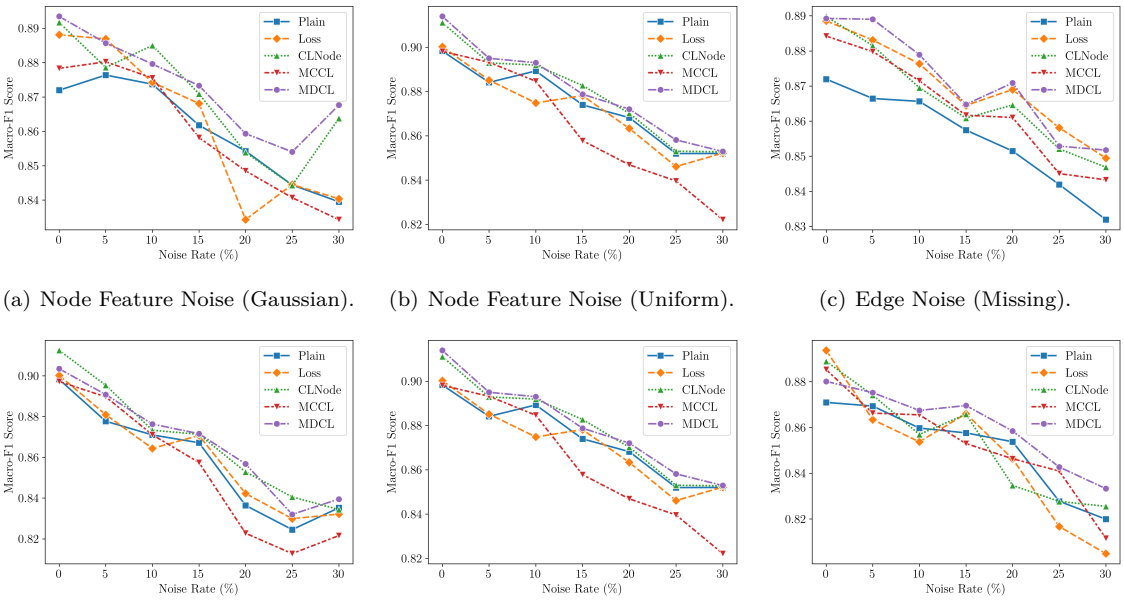
Table 6: Performance (macro-F1) comparison of different difficulty measures on the IMDB dataset using RGCN.

Noise Target	Node Feature		Label		Edge	Mixed
Noise Type	Gaussian	Uniform	Pairwise	Uniform	Missing	
Plain	55.35	56.67	58.65	59.36	62.97	54.77
$+D_{feat}$	<u>57.74</u>	**59.27**	59.37	62.17	62.67	<u>56.44</u>
$+D_{topo}$	56.32	56.68	59.04	62.05	<u>63.16</u>	56.15
$+D_{loss}$	56.58	58.15	**60.05**	**63.63**	63.15	55.66
AvgDiff	55.43	57.75	59.17	62.83	63.15	56.09
MDCL	**58.18**	<u>58.35</u>	<u>59.79</u>	<u>63.25</u>	**63.21**	**56.95**

greater importance for the DBLP and ACM datasets, highlighting the relevance of topological structures and loss-based adjustments in these graphs. This suggests that the models adapt to the specific characteristics of each dataset, emphasizing different aspects of the data. These findings further validate the effectiveness of the adaptive weighting mechanism in handling diverse graph structures and learning tasks.

5 Conclusion

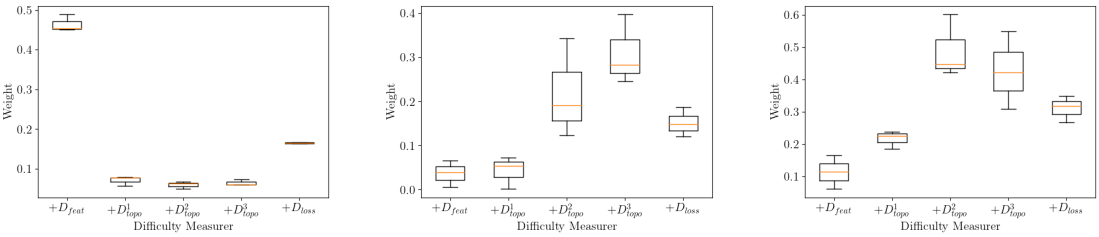
This paper confronts the significant challenge posed by noise in heterogeneous graphs and its detrimental effects on the performance of GNNs. Despite the considerable advancements achieved by existing models and methodologies, their inherent sensitivity to different noise conditions and the complexities associated with heterogeneous data structures frequently undermine their effectiveness. In response to these limitations, we propose a novel approach called Multi-Difficulty Measure Curriculum Learning (MDCL). This method seamlessly integrates multiple difficulty measures specifically designed for heterogeneous graphs, alongside



(a) Node Feature Noise (Gaussian). (b) Node Feature Noise (Uniform). (c) Edge Noise (Missing).

(d) Label Noise (Pairwise). (e) Label Noise (Uniform). (f) Mixed Noise.

Figure 4: Performance comparison of RGCN across varying noise rates and different noise types on the ACM dataset.



(a) IMDB. (b) DBLP. (c) ACM.

Figure 5: Weights learned for different difficulty measures in MDCL on various datasets using RGCN.

an adaptive weighting mechanism that enhances robustness against noise in heterogeneous graphs. By systematically progressing through easier tasks and incorporating nuanced difficulty metrics, MDCL establishes a more resilient training paradigm, which in turn fosters improved accuracy and generalization capabilities in noisy environments. Extensive experimental results provide compelling evidence for the effectiveness of MDCL, showcasing its superiority in managing real-world heterogeneous graph data, particularly in scenarios characterized by high levels and mixed types of noise when compared to existing methodologies. These findings highlight the practical value of MDCL, contributing to the reliability of graph-based learning in noisy, complex settings.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, pp. 41–48, 2009.

Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning*, pp. 695–704, 2019.

Enyan Dai, Charu Aggarwal, and Suhang Wang. NRGNN: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 227–236, 2021.

Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding. In *The Web Conference*, pp. 2331–2341, 2020.

Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R Scott, and Dinglong Huang. CurriculumNet: Weakly supervised learning from large-scale web images. In *European Conference on Computer Vision*, pp. 135–150, 2018.

Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, pp. 8536–8546, 2018.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *The Web Conference*, pp. 2704–2710, 2020.

Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, 22(2):19–34, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

Junnan Li, Richard Socher, and Steven C. H. Hoi. DivideMix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2020.

Xiaohe Li, Zide Fan, Feilong Huang, Xuming Hu, Yawen Deng, Lei Wang, and Xinyu Zhao. Graph neural network with curriculum learning for imbalanced node classification. *Neurocomputing*, 574:127229, 2024.

Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. In *Advances in Neural Information Processing Systems*, pp. 20331–20342, 2020.

Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1150–1160, 2021.

Meng Qu, Jian Tang, and Jiawei Han. Curriculum learning for heterogeneous star network embedding via deep reinforcement learning. In *ACM International Conference on Web Search and Data Mining*, pp. 468–476, 2018.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pp. 593–607. Springer, 2018.

Min Shi, Yufei Tang, Xingquan Zhu, Yuan Zhuang, Maohua Lin, and Jianxun Liu. Feature-attention graph convolutional networks for noise resilient learning. *IEEE Transactions on Cybernetics*, 52(8):7719–7731, 2022.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Nidhi Vakil and Hadi Amiri. Curriculum learning for graph neural networks: a multiview competence-based approach. In *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7036–7051, 2023.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The Web Conference*, pp. 2022–2032, 2019.

Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combating noisy labels by agreement: a joint training method with co-regularization. In *IEEE/CVF Computer Vision and Pattern Recognition Conference*, pp. 13723–13732, 2020.

Xiaowen Wei, Xiuwen Gong, Yibing Zhan, Bo Du, Yong Luo, and Wenbin Hu. CLNode: Curriculum learning for node classification. In *ACM International Conference on Web Search and Data Mining*, pp. 670–678, 2023.

Zhen Hao Wong, Hansi Yang, Xiaoyi Fu, and Quanming Yao. Loss-aware curriculum learning for heterogeneous graph neural networks. *arXiv preprint arXiv:2402.18875*, 2024.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 753–763, 2020.

Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous network representation learning: a unified framework with survey and benchmark. *IEEE Transactions on Knowledge and Data Engineering*, 34(10):4854–4873, 2020.

Hansi Yang, Quanming Yao, Bo Han, and James T. Kwok. Searching to exploit memorization effect in deep learning with noisy labels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–17, 2024.

Xiong Zhang, Cheng Xie, Haoran Duan, and Beibei Yu. NoiseHGNN: Synthesized similarity graph-based neural network for noised heterogeneous graph representation learning. *arXiv e-prints*, pp. arXiv:2412.18267, 2024.

Zheng Zhang, Junxiang Wang, and Liang Zhao. Curriculum learning for graph neural networks: Which edges should we learn first. In *Advances in Neural Information Processing Systems*, pp. 51113–51132, 2024.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2847–2856, 2018.