Experiment & Benchmark Paper: To What Extent Does Quality Matter? The Impact of Graph Data Quality on GNN Model Performance

Jana Vatter
Technical University of Munich
Munich, Germany
jana.vatter@tum.de

Maurice L. Rochau
University of Bayreuth
Bayreuth, Germany
maurice.rochau@uni-bayreuth.de

Ruben Mayer
University of Bayreuth
Bayreuth, Germany
ruben.mayer@uni-bayreuth.de

Hans-Arno Jacobsen
University of Toronto
Toronto, Canada
jacobsen@eecg.toronto.edu

ABSTRACT

Real-world data often is noisy and error-prone which can negatively influence machine learning models. Graph Neural Networks (GNNs) introduce the additional challenge that during training, node information is iteratively passed through the graph along the edges. Consequently, errors or deviations in the graph data could highly impact the model's predictive capability. Our work systematically investigates how quality deviations in graph datasets influence the GNN model performance. We focus on the node features and explore three dimensions: the rate of modified features, the amplitude of modification, and the feature precision. Based on our results, we give insights and recommendations for practitioners. For instance, when using highly clustered graphs, modifying around 40% of the features only results in a slight decrease of performance and the rate of modified features is more crucial than the amplitude of modification. We illustrate practical implications of our results by establishing connections to real world domains such as graph dataset acquisition and efficient GNN training.

VLDB Workshop Reference Format:
Jana Vatter, Maurice L. Rochau, Ruben Mayer, and Hans-Arno Jacobsen. Experiment & Benchmark Paper: To What Extent Does Quality Matter? The Impact of Graph Data Quality on GNN Model Performance. VLDB 2025 Workshop: Large Scale Graph Data Analytics.

1 INTRODUCTION

Graph Neural Networks (GNNs) have gained significant attention in recent years due to their ability to process and learn on graph-structured data effectively. Their versatility has made them applicable to a wide range of domains, ranging from natural language processing [32, 50] and computer vision [5, 23] to recommender systems [12, 20, 47]. As the interest in GNNs continues to grow, a variety of graph datasets [18, 19, 30, 34] have been developed and are commonly adopted for benchmarking, research, and experimentation.

Data encountered in real-world applications is often noisy, error-prone, and incomplete which could (negatively) affect the ML models [11, 15, 21, 36, 40, 46]. This is furthermore exacerbated as data collection itself is a difficult task, and it is hard to get reliable and unbiased data all the time. This holds true across different data collection instruments such as online surveys [35], data mining [26], and online tracking [7]. Considering that most datasets used in academia nowadays are a result of a combination of these data collection instruments, it becomes clear that perfect data is hard to achieve [37]. Further restrictions are the definition of what a perfect dataset constitutes and, in large datasets, the increasing costs to check all data entries for validity [33].

Graphs pose an additional challenge: The graph structure determines the flow of information. During GNN training, information is passed iteratively along edges from the nodes to their neighbors. Consequently, even small deviations and biases in the data could highly influence the training and prediction quality. Errors in the data can be propagated across the graph during the training process which could lead to a decrease in terms of model performance. [43]

In this paper, we therefore focus on the quality of graph-structured data when training GNN models. Our work investigates the impact of varying graph data quality, especially with deviations in the node features and target labels. Through extensive experiments, we aim to quantify the level of imperfection that GNNs can tolerate while maintaining their ability to learn effectively. We explore three dimension: the rate of modified features, the amplitude of modification, and the feature precision. In parallel, we examine common data quality metrics including feature accuracy, target accuracy, and class balance to understand their relationship with model robustness. Our findings provide insights for researchers and practitioners working with GNNs and graph data. We derive practical recommendations on the trade-offs between data quality and resource investment, assessing when and where errors in datasets may be acceptable without significantly compromising model performance. Going further, we connect our findings to application domains including dataset acquisition, and efficient GNN training. By bridging the gap between petridish datasets and real-world graphs, this work contributes to the development of more

Proceedings of the VLDB Endowment. ISSN 2150-8097.

realistic and effective approaches to training and evaluating GNNs in noisy environments.

Our contributions are as follows:

(1) We systematically investigate the impact of perturbed features in the training data on the GNN performance. We systematically increase the number of node features that is modified to determine an upper limit of bias in the data that can be handled by our models. Evaluation is done based on common machine learning metrics such as accuracy, precision, area under the curve, and loss.
(2) We analyze whether graph characteristics such as average node degree and clustering coefficient impact the model when dealing with modified features. Therefore, our experiments include 12 graph datasets with varying average node degree and clustering coefficient. To quantify this, we perform a line of experiments where features of high-degree nodes have a higher probability of being modified.
(3) We investigate the influence of the GNN architecture when using graphs with modified node features. We focus on six commonly used architectures, GCN, GAT, GATv2, SAGE, GIN, GGNN, to assess which impact the architecture has.
(4) We explore the impact of feature precision in terms of decimal places. As rounding errors occur in the real-world, it is important to investigate to which extent our GNN models are affected when trained on less precise data.
(5) We give insights on how our results impact the meaning of good graph data, the acquisition of new graph datasets, and quantization.

2 PRELIMINARIES

In the following, we give an overview of graphs and Graph Neural Networks (GNNs) (Section 2.1), describe the properties of graphs (Section 2.2). We also cover data quality and present selected metrics (Section 2.3).

2.1 Graph Theory and Graph Neural Networks

A graph G captures the relationship between objects. Formally, the graph is denoted by $G = (V, E)$ with the set of vertices V and the set of edges $E \subseteq \{\{u, v\} | u, v \in V\}$. A vertex v depicts an object whereas an edge e indicates a relationship between two vertices.

In order to learn and predict on graph-structured data, a specialized machine learning method has been developed, namely Graph Neural Networks (GNNs) [13, 44]. Alongside neural network (NN) operations such as matrix-multiplication, back-propagation, and weight updates, message passing is performed. A forward pass can be described as follows. First, each vertex is assigned an initial representation. At each training step, each vertex v exchanges and aggregates messages from its neighbors according to an aggregation function

$$a_v^{(l)} = AGGREGATE^{(l)}(\{h_u^{(l-1)} | u \in N(v)\}) \tag{1}$$

where $h_u^{(l-1)} | u \in N(v)$ denotes the activations at layer $l-1$ of the neighbors $N(v)$ [17, 22]. The aggregated messages $a_v^{(l)}$ are then used to update the current representation $h_v^{(l)}$ of vertex v. This can be formalized by

$$h_v^{(l)} = UPDATE^{(l)}(a_v^{(l)}, h_v^{(l-1)}) \tag{2}$$

The aggregation and update steps are repeatedly performed until the given number of layers is reached. After k layers, the k-hop neighborhood is captured. According to a loss function, the model weights are adapted in an NN-like fashion. The subsequent iteration is performed on the updated model.

In the following, we introduce the GNN architectures used in our experiments, namely Graph Convolutional Network (GCN), Graph Attention Network (GAT), SAGEConv, Graph Isomorphism Network (GIN), and Gated Graph Neural Network (GGNN).

Graph Convolutional Network (GCN). The Graph Convolutional Network (GCN) [29] uses a parameter matrix to transform the node representations of the previous layer and assigns weights according to the graph adjacency matrix. The updated node representations can be obtained with $H^{(t+1)} = \sigma(\hat{A}H^{(t)}W^{(t)})$ with the matrix $H^{(t+1)}$ of stacked node representations h_v^t for a node v at the current layer t. σ represents an activation function (e.g., ReLU), $\hat{A}$ is the normalized adjacency matrix A, and there is the parameter matrix W. A shared weight is used for all edges leading to a simple, but expressive model.

Graph Attention Network (GAT). The Graph Attention Network (GAT) [49] assigns different weights to different neighbors in the aggregation function by using masked self-attentional layers. In this way, the model is able to focus on local dependencies. The update function is given through $h_v^{(t+1)} = \sigma(\frac{1}{K}\sum_{k=1}^{K}\sum_{u \in N_v}\alpha_{vu}^k W_k h_u^t)$ where α_{vu}^k is the normalized attention coefficient, k the attention head and W the weight matrix.

GATv2 [3] introduces an attention scoring layer after the activation to rank the attention scores based on the query node. This helps to make the model even more expressive.

SAGE. SAGE [17] is an inductive variant of GCN. The update function is denoted as $h_v^{(t+1)} = \sigma(W^{(t+1)} \cdot CONCAT(h_v^{(t)}, h_{N(v)}^{(t+1)}))$ with a non-linear activation function σ.

Graph Isomorphism Network (GIN). The Graph Isomorphism Network (GIN) [55] is inspired by the Weisfeiler-Lehman (WL) graph isomorphism test and incorporates multi-layer perceptrons (MLPs) to achieve high discriminative power. The nodes are updated via $h_v^{(t+1)} = MLP^{(t+1)}((1+\epsilon^{(t+1)}) * h_v^{(t)} + \sum_{u \in N(v)} h_u^{(t)})$ where ε can be a learnable or fixed parameter. The features are aggregated using a *sum* operator to optimize both expressivity and computational efficiency.

Gated Graph Neural Network (GGNN). A GRU-like gating mechanism for message passing is used by the Gated Graph Neural Network (GGNN) [31] to capture sequential and long-range dependencies. An update step is defined as $h_v^{(t+1)} = GRU(h_v^{(t)}, a_v^{(t+1)})$.

2.2 Statistical Properties of Graphs

Due to the mathematical definition of a graph $G = (V, E)$ (Section 2.1), a regular graph doesn't have any structure or hierarchy beyond the set of nodes V and set of edges E. This also means that a graph can not be ordered. More generally, all vertices and edges exist on

the same plane. This can make it difficult to interpret how individual vertices and edges contribute to a graph on a macro scale.

As the mathematical foundations and methods focus on a micro scale, e.g. updating and aggregating information to immediate neighbors as described in equation 1 and equation 2, it is hard to estimate how these changes affect the overall graph and GNN on a global level. Changing a value in one vertex on one end of the graph might relate to a big, small or even no change in another vertex on the other end of the graph k hops away.

For this reason, we investigate this problem in an experimental way to better understand how the micro-scale methods and foundations affect a graph on a macro level. A related study takes a similar angle at this by randomly deleting nodes in homogeneous graphs [48]. The results are encouraging as up to 40% smaller graphs show the same performance as their complete siblings. This is different to our approach as deletion of information is not the same as a change of information. In graphs, the removal of an edge might, counter-intuitively, add information as the missing connection between two points is an information itself. By systematically altering vertices and features, we aim to explore the behavior of aggregation and updating functions. In this way, we aim to understand the impact of data perturbations during training on the GNN performance; in particular, to what extent data quality degradations can be tolerated before performance collapses.

2.3 Data Quality

Recently, research not only focuses on improving machine learning models, but also highlights the importance of data quality [4, 14, 21, 56]. For that reason, a plethora of data quality metrics has emerged, including `correctness`, `class balance`, `completeness`, and `consistency` [4, 56]. We focus on `correctness` and `class balance` in our experiments as these metrics specifically investigate the dataset features. `Correctness` can be distinguished into two categories: `Feature Accuracy` and `Target Accuracy`. The following definitions are oriented on Budach et al. [4].

Feature Accuracy. The feature accuracy metric indicates to which extent the feature values deviate from their correct values. For numerical values, it is given by

$$FeatureAccuracy(d) = \frac{1}{n_{num}} * \sum_{i=0}^{n_{num}-1} FAcc(c_i) \tag{3}$$

where d is the dataset, n_{num} the number of numerical features, and c_i a feature. $FAcc(c_i)$ is denoted by

$$FAcc(c_i) = 1 - \frac{avg\_dist(c_i)}{mean\_gt(c_i)} \tag{4}$$

with the average absolute distance $avg\_dist(c_i)$ between the ground truth and the given values, and the mean value of the ground truth $mean\_gt(c_i)$.

Target Accuracy. In contrast to feature accuracy, the target accuracy measures how accurate the target labels that should be learned and predicted by the model are compared to the ground truth. The target accuracy for categorical values is the ratio of correct target values with respect to the ground truth.

$$TargetAccuracy(d) = 1 - \frac{mismatches(target)}{n} \tag{5}$$

where $mismatches(target)$ denotes the number of incorrect target features, n is the total number of samples.

Target Class Balance. Many ML methods benefit from relatively balanced numbers of samples per class. The presence of a large majority class can lead to overfitting on this class. Therefore, it is important to explore the target class balance. The *Balance* can be described through

$$Balance(d) = 1 - \frac{ImBalance(d)}{\epsilon} \tag{6}$$

$$ImBalance(d) = \frac{1}{2} * \sum_{i,j \in 1,\dots,m} |n_{cl_i} - n_{cl_j}| \tag{7}$$

where $\epsilon = \lceil \frac{m}{2} \rceil \cdot \lfloor \frac{m}{2} \rfloor \cdot n_{cmax}$, m is the number of target classes, n_cmax the maximum number of samples a class can have, and n_{cl_x} is the number of samples in class x.

3 EXPERIMENTAL METHODOLOGY

This section presents our experimental methodology, including the datasets we use (Section 3.1), how we modify the features (Section 3.2) and the GNN architectures we experiment with (Section 3.4). As a framework, we use the well-known Deep Graph Library (DGL) [9, 51] in combination with PyTorch [39].

3.1 Datasets

For our experiments, we use 12 publicly available graphs of sizes ranging from 2,700 nodes to over 2,4 million nodes (Table 1). The graphs are either available in the Open Graph Benchmark (OGB) [19, 38] or in the Deep Graph Library (DGL) [9, 51]. The graphs depict various domains including scientific citations, product co-purchasing, and forum social networks. Consequently, their characteristics in terms of average node degree (ND) and average clustering coefficient (CC) also differ. In this manner, we ensure a wide variety of graph datasets are included. We use the node features and train/validation/test splits that are provided with the datasets.

3.2 Feature Modification

Input and output features (ground truth labels) are modified alike. We modify the features across two dimensions: the rate of modified features (RMF) and the amplitude of modification (AM). RMF illustrates the percentage of features that is modified, while AM descripes the degree of modification. For AM, we go in steps of 20% and for RMF, in steps of 10%. Let's assume we modify 10% of the features (RMF) up to 10% (AM). First, we select 10% of all features to be modified. The values of these 10% of all features are modified by up to ±10% meaning some feature values are modified more than others. This procedure ensures there is no systematic data drift. In contrast to the input features, the output features are the classes we want to predict, i.e., we deal with categorical data. Here, instead of modifying by up to ±10%, we randomly select one of the possible prediction classes. It should be noted that we modify the set of training nodes only. The test nodes are left untouched.

3.3 Degree-aware Feature Modification

In addition to randomly selecting the features that are modified, we introduce a line of experiments where the features are perturbed

Table 1: Characteristics of the datasets based on [19] (ND: average node degree, CC: average clustering coefficient)

Name	#Nodes	#Edges	ND	CC	#Classes	Feature Size	Domain
Cora	2,708	10,556	3.9	0.241	7	1,433	citation network
CiteSeer	3,327	9,228	2.8	0.141	6	3,703	citation network
AmazonPhoto	7,650	238,163	31.1	0.404	8	745	product co-purchasing
AmazonComputer	13,752	491,722	35.8	0.344	10	767	product co-purchasing
CoAuthorCS	18,333	163,788	8.9	0.433	15	6,805	citation network
PubMed	19,717	88,651	4.5	0.06	3	500	citation network
CoAuthorPh	34,493	495,924	14.8	0.378	5	8,415	citation network
Flickr	89,250	899,756	10.1	0.033	7	500	image network
arxiv	169,343	1,166,243	13.7	0.226	40	128	citation network
Reddit	232,965	114,615,892	492	0.579	50	602	forum social network
Yelp	716,847	13,954,819	20.5	0.092	100	300	reviews about businesses
products	2,449,029	61,859,140	50.5	0.411	47	100	product co-purchasing

based on the node degree. Higher probability to be perturbed is given to node features belonging to high-degree nodes. Then, we use a random choice with weights to select the node features. In this way, we investigate whether high-degree nodes have a higher influence on the overall GNN performance since they have more neighbors to which perturbed features are propagated to.

3.4 GNN architecture

The Graph Convolutional Network (GCN) [29], the Graph Attention Network (GAT) [49], GATv2 [3], GraphSAGE [17], Graph Isomorphism Network (GIN) [55], and Gated Graph Neural Network (GGNN) [31] are used for our experiments with two layers, a learning rate of 0.001 and the Adam optimizer [28]. Mini-batch training with random neighbor sampling [17] is performed for 20 epochs with sample size 10 and each experiment is repeated 3 times with 3 different random seeds. This means we create and train on three modified graphs for each experiment configuration. The prediction task is node classification.

4 EMPIRICAL EVALUATION

We cluster the graphs into categories depending on the graph characteristics and results. In the following, we present the results of one representative of each category. `Cora`, `CiteSeer`, and `PubMed` share a node degree (ND) between 2.8-5 and low clustering coefficient (CC) with 0.06-0.25. An ND of 8-15 and CC of 0.3-0.44 can be observed for `CoAuthorCS`, `CoAuthorPh`, and `Flickr`. Graphs with an even higher ND (34-36), but similar CC (0.3-0.41) are `AmazonComputer` and `AmazonPhoto`. The two graphs with the highest ND (50-500) and CC (0.41-0.6) are `Reddit` and `products`. The remaining graphs are `arxiv` (ND: 13.7, CC: 0.226) and `Yelp` (ND: 20.5, CC: 0.092).

4.1 Model performance

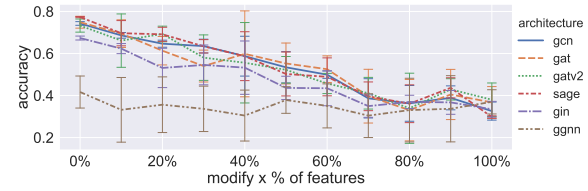
This section presents our results for experiments on the model performance. First, we vary the number of features that are modified and explore three GNN architectures (Section 4.1.1). After that, we show how the degree of modification influences the models (Section 4.1.2) and investigate the impact of the feature precision in terms of decimal places (Section 4.1.4).

4.1.1 Rate of Modified Features. For the following experiments, we vary the rate of features that are modified (RMF) from 0-100% in steps of 10%, the amplitude of modification remains 20%. We use six GNN architectures, namely GCN, GAT, GATv2, SAGE, GIN, and GGNN.

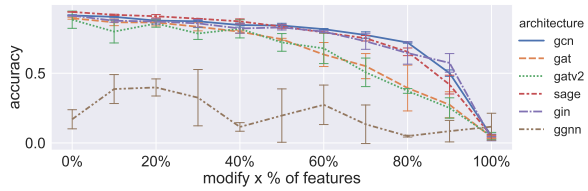
In Figure 1, we summarize our findings. For graphs with low ND and low CC, such as `PubMed` (Figure 1a), the accuracy linearly decreases with increasing RMF. Since each node only has a few connections to neighboring nodes, diverging features highly impact the model training and following prediction task. For graphs with medium to high ND and CC (`CoAuthorCS`, `arxiv`, `products`), the performance of the model only slightly decreases with a higher number of modified features. For instance, the accuracy for `CoAuthorCS` (Figure 1b) only decreases $\sim$ 5% when modifying 50% of features. Since each node has numerous connections, the impact of a corrupted neighbor is not as high compared to nodes with only few connections.

It is interesting that the accuracy when using GCN, SAGE, or GIN, usually does not decrease as fast as for GAT and GATv2. Further, the curve for GAT and GATv2 is not as stable as for most other architectures. An explanation is that GAT and GATv2 both use the attention mechanism within their aggregation function. Besides global features, these architectures focus on local features. Consequently, a modified feature has a much higher impact on the overall performance compared to architectures such as GCN leading to a decrease of accuracy and less stability with a higher degree of modification.

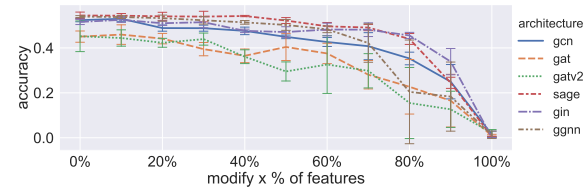
4.1.2 Amplitude of Modification. In the following, we present the results for different amplitudes of modification (AM) ranging from 0-100% in steps of 20% and RMF from 0-100% in steps of 10%. The x-axis shows the RMF, the y-axis the accuracy, and the different linestyles represent the different AM values. Here, we are using the GCN as architecture. Across all datasets (Figure 2), the accuracy only slightly differs when transitioning from 0% to 100% of modification. Our experiments show that when training and applying GNNs, not only the node features are important, but also the graph structure. How nodes relate to each other, also across multiple hops, plays a large role for GNNs. Although some node features are highly



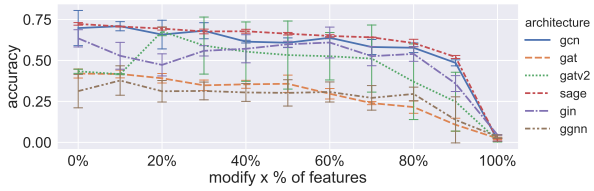
(a) PubMed



(b) CoAuthorCS

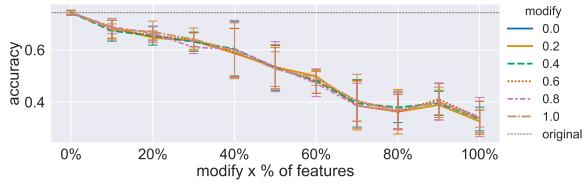


(c) ogbn-arxiv

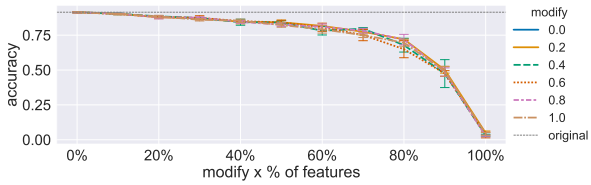


(d) ogbn-products

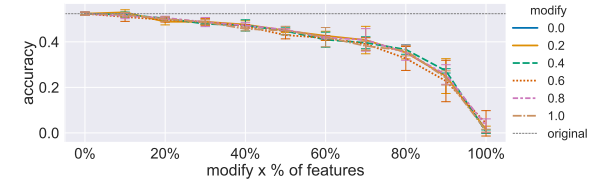
Figure 1: Accuracy for different datasets, architectures, and rate of modified features (RMF)



(a) PubMed



(b) CoAuthorCS



(c) ogbn-arxiv

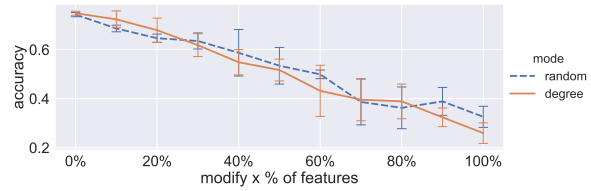
(d) ogbn-products

Figure 2: Accuracy for different datasets, rate of modified features (RMF), and amplitude of modification (AM) when using GCN

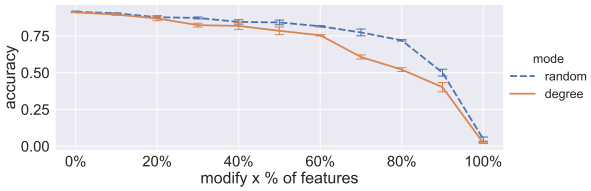
modified, the structure and relations are preserved leading to a stable accuracy.

4.1.3 Degree-aware Feature Modification. In the following, we present our results for GCN when assigning a higher probability of modification to node features of high-degree nodes (Figure 3). Interestingly, the accuracy does not decrease significantly faster when modifying up to 40% of features for most datasets. With more than 40% of modified node features, we can see that the degree-aware modification usually performs slightly worse compared to random modification. This shows that the node feature quality of high-degree nodes can influence the GNN more than features of lower-degree nodes. However, this behavior only occurs after a certain percentage of perturbed node features is surpassed. This further supports our findings that not only the quality of features impacts GNN performance, but also the the graph structure and global relations.

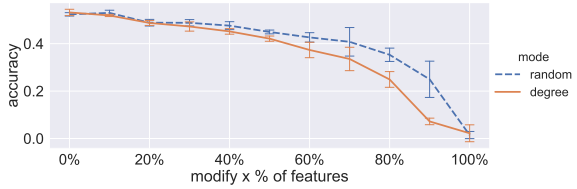
4.1.4 Feature Precision. Not only divergent feature values can occur in the real world, but also reduced precision in terms of decimal places. This can have various reasons: rounded floating point values or less precise measurements. In our datasets, the maximum number of decimal places in the feature tensors ranges from 30 to 50. Therefore, we experiment with less decimal places, from 0 to 32 and compare it to the full precision. In addition to measuring the model's performance, we also measure the data quality in terms of Feature Accuracy. Figure 4 shows the results for the `PubMed`, `ogbn-arxiv`, and `ogbn-products` graph. The accuracy for `PubMed` increases along with the Feature Accuracy when using a larger number of decimal places for the features. The model accuracy even increases slightly faster than the Feature Accuracy. With a precision of 2 decimal places, there is no significant difference between the model accuracy and the accuracy when using full precision. A similar behavior can be seen for `ogbn-arxiv`. The curve



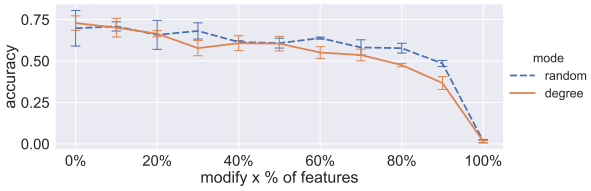
(a) PubMed



(b) CoAuthorCS

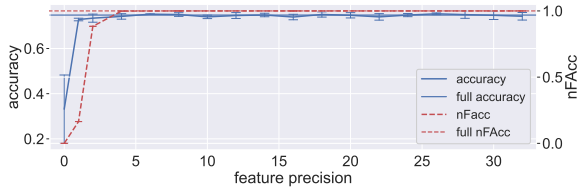


(c) ogbn-arxiv

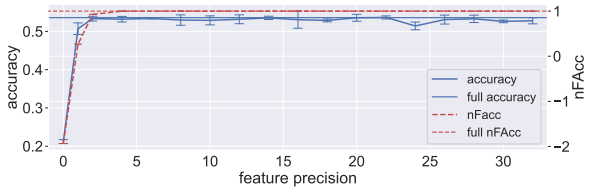


(d) ogbn-products

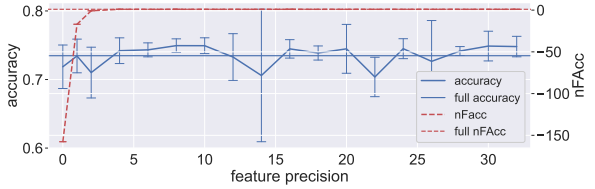
Figure 3: Accuracy for different datasets, rate of modified features (RMF), and degree-aware modification when using GCN



(a) PubMed



(b) ogbn-arxiv



(c) ogbn-products

Figure 4: Varying Feature Precision for different datasets

for `ogbn-products` is slightly different. It is unstable until a precision of 8 decimal places. The difference of `ogbn-products` to other datasets is the higher ND and CC. With more clusters and a lot of connections within a graph, precision plays a higher influence.

Analogous to our previous experiments, we show that graph data quality is manifold and is the interaction of multiple criteria.

4.2 Data Quality

To better understand the results and to draw connections between prediction performance and data quality, we measure `Feature Accuracy`, `Target Feature Accuracy`, and `Target Class Balance`. We evaluate all datasets and different RMF, ranging from 0% to 100% of modified features in steps of 10% as well as 0% to 100% AM in steps of 20%. We show the results for the `CoAuthorCS` graphs in Figure 5. The results are supported by the remaining datasets which are analogue to `CoAuthorCS`.

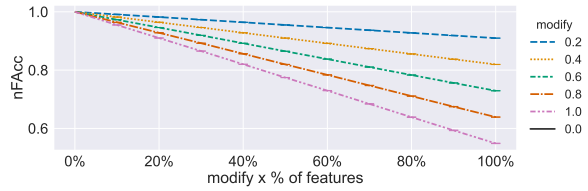
The Feature Accuracy (Figure 5a) decreases linearly with a higher RMF. Whenever the features are modified to a higher amplitude (AM), the Feature Accuracy also is lower compared to a lower amplitude of modification. One could expect to also see a linear decrease in terms of accuracy in the experimental results and differences with varying degrees of modification (Figure 2b). Surprisingly, this does not hold true for all datasets. We can only see a nearly linear decrease with `PubMed`, but there is no significant difference between the amplitude of modification (AM). This further supports our hypothesis that the connections within a graph play a crucial part when training GNNs, and not only the feature quality.

When investigating the Target Accuracy (Figure 5b), we notice a similar behavior. While the Target Accuracy decreases linearly, the accuracy of the predictions does not decrease linearly. Until $\tilde{6}0\%$ of modification (RMF), the values only decrease very slightly, after that the accuracy usually drops.

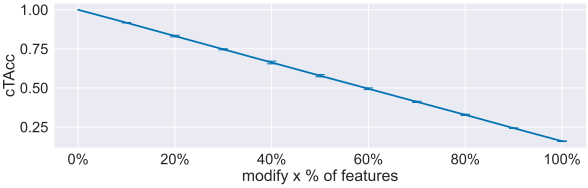
It is interesting that the Target Class Balance (Figure 5c) increases with a higher percentage of modified features (RMF). A reason for this is that with randomly modified target features, each class gets chosen with the same probability leading to a more equal class distribution.

5 KEY INSIGHTS

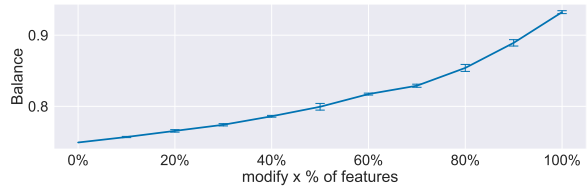
This section explores the implications our findings have on real world domains. We draw connections between our experimental



(a) Feature Accuracy



(b) Target Accuracy



(c) Balance

Figure 5: Data quality metrics for CoAuthorCS

results and various domains, including data quality, dataset acquisition, quantization and differential privacy.

5.1 How do data quality and performance relate to each other?

We have shown that data quality does not necessarily reflect the performance of the GNN model. While data quality gives a proxy to assess the performance, there are more factors to consider. An example is that a modification of around 50% in the `CoAuthorCS` dataset leads to only around 5% loss in terms of accuracy. Only after that, performance starts to drop more significantly. Across all experiments we show that between 40% to 50% of node features can be altered while preserving adequate performance. This has a clear practical implication: Data quality is important but GNNs can still deliver sufficient performance on graphs with noisy data. Certain GNN architectures, like SAGE and GCN, are better suited for this than others.

In our second row of experiments where we modified node features either randomly or by node degree, we show that there is no significant difference in terms of performance with up to 40% of modified features. After that, the performance of degree-based modification decreases faster compared to random perturbations.

Our findings indicate that to a certain extent, global factors can cover for lacking data quality on a node level. Consequently, graph data quality is inherently multifaceted, encompassing not only node features and labels, but also global structural properties that are hard to capture. This suggests that practitioners working with real-world noisy graph data should not be discouraged by possible imperfections in their datasets, as GNNs can often extract meaningful patterns despite quality limitations.

5.2 Graph dataset acquisition

Acquiring new datasets can be a resource-consuming process. The goal is to get a good representation of the real world. Why not perfect? For most use cases, it is practically infeasible to have near-perfect data points due to bias in measurement instruments or uncertainty in actual data collection (what is the data we actually need?). Our experiments support the hypothesis that there is a trade-off between the model's performance and the expenditure of dataset acquisition. This is supported by two findings: 1) To a certain degree, the GNN model is able to handle deviations in the features. 2) Less feature precision influences the GNN model's performance, but for our datasets, 4 decimal places are sufficient to achieve similar results compared to the original precision. These findings lead to the theory that training on faulty or biased graph data (to a certain degree) does not influence the GNN model as crucially as expected and might be negligible as performance is only affected for very stark deviations. For data acquisition, this would mean that certain inaccuracies can be tolerated which could make overall data capturing simpler.

According to our experiments, data acquisition should be less focused on collecting overly perfect datasets but more focused on trying to collect the connections between data points.

5.3 Feature precision in GNN training

Graphs can get extremely large with over billions of nodes and edges [16, 18]. Consequently, methods to train efficiently on those graphs are needed. Using lower precision for graph features, weights, and parameters is one of these methods. In this way, one can achieve fast and memory-efficient training [6, 10, 52]. When using less precision, valuable information might be lost. Therefore, some approaches use sophisticated compression techniques [10], modification of the message propagation scheme [52], as well as binarized representations and bit-wise operations [6].

We show that to a certain degree, we can use less precision in the graph data without using a specialized method with almost no decrease in terms of predictive performance. The key question is: How much precision is truly necessary? If we were to predict a person's height at the age of 18, is giving the height centimeters enough, or do we want to have millimeters or even micrometers? If we have the height in millimeters, is this really superior to having only centimeters or do we just discard the additional precision in favor of practicality anyway? In conclusion, it is most important to choose the right precision for the right question and not just add more decimal places.

For our experiments, we show that in some experiments only 4 decimal places are sufficient. In PyTorch, a popular Python Machine Learning framework, the standard data type is float32. By using 4 decimal places instead of 32, the precision is 8 times lower, and we still achieved satisfactory results. For practitioners of Graph Machine Learning frameworks this means that it might be a good option to use significant less precision as it can deliver comparable model performance.

6 RELATED WORK

In their work, Renggli et al. [40] provide a general, data-centric view of MLOps. They propose a framework CPClean which analyzes the impact of noisy data on the corresponding general ML model to clean the data in relation to the gained insight. Northcutt et al. [37] investigate label errors in commonly used benchmark datasets in the fields of computer vision, natural language, and audio datasets. They identify and quantify errors in the datasets and their impact on ML model performance. Jain et al. [21] highlight the importance of data quality for ML in general. They present data quality metrics and show how to transform the data to address quality gaps. Conversely, our work is focused on data quality of graphs and their impact on GNN training.

Zügner et al. [57] perform adversarial attacks on graphs, meaning certain information is manipulated. Here, the authors modify the graph structure and features. Their objective is to make the changes as unnoticeable as possible. For instance, they change features but aim to preserve the feature statistics. This is contrary to our work where we focus on noticeably modifying graph features and measuring the influence on GNN models. Wu et al. [54] investigate adversarial attacks leading to incorrect predictions. They focus on gradient-based adversarial attacks and explore how such an attack can be performed and how to defend it. We distinguish ourselves by exploring modified graph features. Pro-GNN [25] proposes a novel strategy to defend adversarial attacks. The graph structure is learned alongside the model parameters given a pertubed graph. In this way, robustness is ensured. Jin et al. [24] concentrate on adversarial attacks that concern the edges. The authors give an overview of existing attacks and defenses. Our work differs by focusing on perturbations of node features.

There is also work on robust GNN training. For instance, the framework NIFTY [2] can be used to achieve, fair and stable GNN training. To ensure stability during training, the similarity between representations of perturbed graphs (node features and graph structure) and a counterfactual graph are maximized. This leads to overall stable graph representations and predictions. Dai et al. [8] propose a robust structural noise-resistant GNN (RS-GNN) framework. First, a graph with structure noise and missing labels is used to generate a dense, denoised graph. Alongside the original graph, the denoised graph is trained on to improve the predictions on unlabeled nodes. While the framework has been proven to work well, it only addresses structural noise and not attribute noise. The *random* GCN [45] enhances the message passing step by adding a node feature kernel. The GCN performance on perturbed graphs is improved and the model is more robust. Wang et al. [53] augment sets of nodes which are trained alongside the original ones to strengthen the consistency of GCN models. It is shown that semi-supervised node classification can be improved with their data augmentation method. Aburidi et al. [1] use meta-gradients for robust GNN training. The graph structure is regarded as hyperparameter which needs to be optimized. In this way, GNNs can be trained in a robust way. We distinguish ourselves by exploring attribute perturbations instead of structural perturbations.

7 CONCLUSIONS

Real-world graph data often is noisy and error-prone. Therefore, we investigate how deviations of benchmark datasets influence the performance of GNN models by modifying node features and labels. We explore the impact of the rate of modified node features (RMF) and the amplitude of modification (AM) by using various graphs of different characteristics and size. We derive the rule that with GCN, around 40% of features can be modified while maintaining a stable accuracy. The degree of modification is not as influential as the number of modified features. Further, we vary the number of decimal places of the node features to simulate reduced precision. For our datasets, 2 decimal places usually are sufficient for an acceptable accuracy. In addition, we establish connections to application domains such as dataset acquisition and using lower precision for training.

Future Work. Our work focuses on homogeneous graphs. However, there are many different graph types, such as heterogeneous or temporal graphs which pose different challenges. Heterogeneous graphs can consist of multiple node and edge types. One could investigate whether errors only concerning certain types influence the GNN model. Temporal graphs change over time. Here, deviations in the data are not only potentially propagated across the graph, but also possibly passed across multiple time steps which could highly affect the model.

As graphs are all around us, they can also include sensitive data. Depending on the use case, one might want to learn and predict on this kind of data in a privacy preserving manner. One way to achieve this is by employing differential privacy (DP). Multiple approaches have emerged, from adding noise to the aggregation function [42] and training submodels [41] to perturbation of the graph structure by replacing similar nodes [27]. Another work states that random sparsification of large-scale graphs can be done with deleting up to 60% of the nodes without experiencing a large decrease in terms of accuracy [48]. Combined with our findings of random feature modification, this highlights the usability of DP with random modifications and opens new opportunities for DP research.

With our experiments showing that graph data can be altered considerably while still achieving satisfactory performance, the question arises how this behavior can be described in a generalized way for the whole graph or GNN models, namely the macro level. It is currently unclear how graphs and their characteristics can be described concisely on a global level. By changing data, it is expected that performance should degrade as vertex information is lost and it becomes harder to achieve convergence. However, we observe the opposite where a decrease of data quality only leads to a slight drop in terms of model performance up to a certain degree ($\tilde{4}0\%$ of modified features). After that, the performance usually collapses. Therefore, future research could focus on the formal description of graphs on the macro-level.

Current research in non-graph based Machine Learning suggest that having lower quality data leads to performance losses in classification and regression tasks [4, 15, 21]. The key difference between classical methods and graph-based ones needs to be investigated further alongside how data alterations change model performance. Additionally, if decreased data quality does not impact graph-based methods as much as traditional ones, the question arises if graph-based approaches should be favored in situations where traditional methods struggle.

REFERENCES

[1] Mohammed Aburidi and Roummel Marcia. 2024. Topological adversarial attacks on graph neural networks via projected meta learning. In *2024 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS)*. IEEE, 1–8.

[2] Chirag Agarwal, Himabindu Lakkaraju, and Marinka Zitnik. 2021. Towards a unified framework for fair and stable graph representation learning. In *Uncertainty in Artificial Intelligence*. PMLR, 2114–2124.

[3] Shaked Brody, Uri Alon, and Eran Yahav. 2022. How Attentive are Graph Attention Networks?. In *International Conference on Learning Representations*.

[4] Lukas Budach, Moritz Feuerpfeil, Nina Ihde, Andrea Nathansen, Nele Noack, Hendrik Patzlaff, Felix Naumann, and Hazar Harmouch. 2022. The effects of data quality on machine learning performance. *arXiv preprint arXiv:2207.14529* (2022).

[5] Chaoqi Chen, Yushuang Wu, Qiyuan Dai, Hong-Yu Zhou, Mutian Xu, Sibei Yang, Xiaoguang Han, and Yizhou Yu. 2024. A survey on graph neural networks and graph transformers in computer vision: A task-oriented perspective. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[6] Yankai Chen, Huifeng Guo, Yingxue Zhang, Chen Ma, Ruiming Tang, Jingjie Li, and Irwin King. 2022. Learning binarized graph representations with multi-faceted quantization reinforcement for top-k recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 168–178.

[7] Clara Christner, Aleksandra Urman, Silke Adam, and Michaela Maier. 2022. Automated tracking approaches for studying online media use: A critical review and recommendations. *Communication methods and measures* 16, 2 (2022), 79–95.

[8] Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. 2022. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 181–191.

[9] DGL Team. 2024. Deep Graph Library. https://www.dgl.ai/.

[10] Mucong Ding, Kezhi Kong, Jingling Li, Chen Zhu, John Dickerson, Furong Huang, and Tom Goldstein. 2021. VQ-GNN: A universal framework to scale up graph neural networks using vector quantization. *Advances in Neural Information Processing Systems* 34 (2021), 6733–6746.

[11] Wenfei Fan, Ping Lu, Chao Tian, and Jingren Zhou. 2019. Deducing certain fixes to graphs. *Proceedings of the VLDB Endowment* 12, 7 (2019), 752–765.

[12] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems* 1, 1 (2023), 1–51.

[13] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Vol. 2. IEEE, 729–734.

[14] Venkat Gudivada, Amy Apon, and Junhua Ding. 2017. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software* 10, 1 (2017), 1–20.

[15] Nitin Gupta, Shashank Mujumdar, Hima Patel, Satoshi Masuda, Naveen Panwar, Sambaran Bandyopadhyay, Sameep Mehta, Shanmukha Guttula, Shazia Afzal, Ruhi Sharma Mittal, et al. 2021. Data quality for machine learning tasks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 4040–4041.

[16] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Zadeh. 2013. Wtf: The who to follow service at twitter. In *Proceedings of the 22nd international conference on World Wide Web*. 505–514.

[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems* 30 (2017).

[18] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. 2021. OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs. *NeurIPS* 34 (2021).

[19] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems* 33 (2020), 22118–22133.

[20] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. Mixgcf: An improved training method for graph neural network-based recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 665–674.

[21] Abhinav Jain, Hima Patel, Lokesh Nagalapatti, Nitin Gupta, Sameep Mehta, Shanmukha Guttula, Shashank Mujumdar, Shazia Afzal, Ruhi Sharma Mittal, and Vitobha Munigala. 2020. Overview and importance of data quality for machine learning tasks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3561–3562.

[22] Zhihao Jia, Sina Lin, Mingyu Gao, Matei Zaharia, and Alex Aiken. 2020. Improving the accuracy, scalability, and performance of graph neural networks with roc. *Proceedings of Machine Learning and Systems* 2 (2020), 187–198.

[23] Licheng Jiao, Jie Chen, Fang Liu, Shuyuan Yang, Chao You, Xu Liu, Lingling Li, and Biao Hou. 2022. Graph representation learning meets computer vision: A survey. *IEEE Transactions on Artificial Intelligence* 4, 1 (2022), 2–22.

[24] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. 2021. Adversarial Attacks and Defenses on Graphs. *SIGKDD Explor. Newsl.* 22, 2 (Jan. 2021), 19–34. https://doi.org/10.1145/3447556.3447566

[25] 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.

[26] Sethunya R Joseph, Hlomani Hlomani, and Keletso Letsholo. 2016. Data mining algorithms: an overview. *Neuroscience* 12, 3 (2016), 719–43.

[27] Rucha Bhalchandra Joshi, Patrick Indri, and Subhankar Mishra. 2024. GraphPrivatizer: Improved Structural Differential Privacy for Graph Neural Networks. *Transactions on Machine Learning Research* (2024).

[28] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations* (2015).

[29] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[30] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[31] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. 2016. Gated Graph Sequence Neural Networks. In *International Conference on Learning Representations*.

[32] Bang Liu and Lingfei Wu. 2022. Graph neural networks in natural language processing. *Graph Neural Networks: Foundations, Frontiers, and Applications* (2022), 463–481.

[33] Stefano Marchesin and Gianmaria Silvello. 2024. Efficient and Reliable Estimation of Knowledge Graph Accuracy. *Proceedings of the VLDB Endowment* 17, 9 (2024), 2392–2403.

[34] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663* (2020).

[35] MSDP Nayak and KA Narayan. 2019. Strengths and weaknesses of online surveys. *technology* 6, 7 (2019), 0837–2405053138.

[36] Curtis Northcutt, Lu Jiang, and Isaac Chuang. 2021. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research* 70 (2021), 1373–1411.

[37] Curtis G Northcutt, Anish Athalye, and Jonas Mueller. 2021. Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

[38] OGB Team. 2024. Open Graph Benchmark. https://ogb.stanford.edu/.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32 (2019).

[40] Cedric Renggli, Luka Rimanic, Nezihe Merve Gürel, Bojan Karlaš, Wentao Wu, and Ce Zhang. 2021. A Data Quality-Driven View of MLOps. *IEEE Data Engineering Bulletin* 44, 1 (2021), 11–23.

[41] Sina Sajadmanesh and Daniel Gatica-Perez. 2024. Progap: Progressive graph neural networks with differential privacy guarantees. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 596–605.

[42] Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez. 2023. {GAP}: Differentially Private Graph Neural Networks with Aggregation Perturbation. In *32nd USENIX Security Symposium (USENIX Security 23)*. 3223–3240.

[43] Rubab Zahra Sarfraz. 2024. Towards Semi-Supervised Data Quality Detection in Graphs. *Proceedings of the VLDB Endowment. ISSN* 2150 (2024), 8097.

[44] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.

[45] Mohamed El Amine Seddik, Changmin Wu, Johannes F Lutzeyer, and Michalis Vazirgiannis. 2022. Node feature kernels increase graph convolutional network robustness. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 6225–6241.

[46] Nabeel Seedat, Nicolas Huynh, Fergus Imrie, and Mihaela van der Schaar. 2024. You can't handle the (dirty) truth: Data-centric insights improve pseudo-labeling. *Journal of Data-centric Machine Learning Research* (2024).

[47] Kartik Sharma, Yeon-Chang Lee, Sivagami Nambi, Aditya Salian, Shlok Shah, Sang-Wook Kim, and Srijan Kumar. 2024. A survey of graph neural networks for social recommender systems. *Comput. Surveys* 56, 10 (2024), 1–34.

[48] Jana Vatter, Maurice L Rochau, Ruben Mayer, and Hans-Arno Jacobsen. 2024. Size Does (Not) Matter? Sparsification and Graph Neural Network Sampling for Large-scale Graphs. *Proceedings of the VLDB Endowment. ISSN* 2150 (2024), 8097.

[49] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[50] Kunze Wang, Yihao Ding, and Soyeon Caren Han. 2024. Graph neural networks for text classification: A survey. *Artificial Intelligence Review* 57, 8 (2024), 190.

[51] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).

[52] Shuang Wang, Bahaeddin Eravci, Rustam Guliyev, and Hakan Ferhatosmanoglu. 2023. Low-bit quantization for deep graph neural networks with smoothness-aware message propagation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2626–2636.

[53] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. 2020. Nodeaug: Semi-supervised node classification with data augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 207–217.

[54] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples for graph data: deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 4816–4823.

[55] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.

[56] Yuhan Zhou, Fengjiao Tu, Kewei Sha, Junhua Ding, and Haihua Chen. 2024. A Survey on Data Quality Dimensions and Tools for Machine Learning Invited Paper. In *2024 IEEE International Conference on Artificial Intelligence Testing (AITest)*. IEEE, 120–131.

[57] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2847–2856.