TABLE OF CONTENTS

Association for Computing Machinery

Advancing Computing as a Science & Profession

An Interview with Professor Jure Leskovec, Recipient of ACM SIGKDD 2023 Innovation Award

The ACM SIGKDD Innovation Award is conferred annually for outstanding technical innovations that have significantly influenced research and development in the field of knowledge discovery and data mining. 2023 Innovation Award recipient Dr. Jure Leskovec is professor of computer science at Stanford University and co-founder of Kumo.ai, a company bringing the most powerful graph learning approaches to relational data.

First, congratulations on your selection as the 2023 ACM SIGKDD Innovation Award honoree. What does this recognition mean to you?

Thank you, it's such an honor to be selected for the SIGKDD Innovation Award and it truly came as a surprise. I'm humbled by this recognition from academic and industry peers. Perhaps most gratifying is that the award shows appreciation for the impact of a given research area. And the award is not just for me, but it also for the excellent students and collaborators, without whose hard work and dedication none of this would be possible.

Would you please share a bit about yourself and your background?

As professor of computer science at Stanford, my general research area is applied machine learning for large interconnected systems, focusing on modeling complex, richly labeled relational structures, graphs, and networks. Applications of our work include reasoning, recommender systems, computational social science, and computational biology with an emphasis on drug discovery.

After completing my undergraduate studies in Slovenia, I moved to the U.S. to pursue my Ph.D. at Carnegie Mellon University, where I primarily explored graphs in social media. At the conclusion of my graduate program, I was fortunate enough to spend a year at Cornell and then landed at Stanford in 2009. I've always enjoyed working on hard practical problems, so over the years had great collaborations with industry from Facebook and LinkedIn to Twitter, Amazon and Pinterest, where I served as Chief Scientist for 7 years and built an entire family of AI capabilities that transformed the company and its business.

Thinking back on your early interests, when did you decide to go into the field of data science and machine learning?

Computers are these amazing machines that do useful work for humans. So, a machine that can learn from data is an ultimate instantiation of this. I always liked not just the engineering aspect of machine learning but also the angle of data science, where one uses data and computation to make new discoveries about the world around us. So, using computation to discover new knowledge, something that was yet unknown about the world, is the ultimate challenge and this has been fascinating to me ever since I started working with computers.

Admittedly, I never really planned but rather followed my curiosity. I get bored easily and always try to ask different questions. So, it's been a very natural, organic progression to where I am today.

It was also shaped by many dedicated mentors as well as various opportunities that life has brought over the years.

What does innovation mean to you?

At a high level, I view innovation as the desire to walk a path no one has walked before and doing so without a map or signposts along the way. It's an iterative process – we take steps, learn about their effectiveness, decide to progress or adjust course, all without losing sight of the bigger goal in front of us.

As I'm fortunate to have my work in computer science and data science span both academia and industry, I have two views of innovation. From the academic perspective, the objective of explore and show what is still possible and how the future might look like. In the industrial or applied engineering world, the goal of innovation is about solving a real-world problem by doing something better, faster or more accurately than before. In this realm, it's important to understand how to manage – or balance – the quest for innovation versus risk, asking ourselves what is the goal, what is still possible and what are the risks.

I tend to view innovation not as a goal but as a consequence of asking the right questions and then acting on them. When we land on the right question, it's as though we've crossed the mountain peak to find a previously unknown valley, with sunlight streaming in. As a researcher, I'm continuously surprised to learn of the (figurative) diversity of flora and fauna in these previously unexplored valleys. New way of thinking emerges and we amazing research is possible. By continuing to ask questions, we're more apt to experience these amazingly gratifying moments of discovery.

Can you share examples of how innovation manifests in your work?

I feel fortunate to have experienced several discoveries in my career, which is quite humbling. The very first paper of mine published by KDD was one such example of the discovery of a natural law in evolution of graphs. Numerous questions emerged: Is what I've found interesting? Is it contrary to what others think should be? Is my analysis wrong? Is the finding repeatable? These and other questions continue to shape my approach to innovation today.

In the early stages of the Covid 19 pandemic, I led a research project leveraging cell phone mobility data to better understand Covid transmission patterns. An initial question asked was, "Can we build a tool to predict more accurately how Covid will spread?" But we didn't stop there. We continued to push forward with additional questions: Which places are the most infectious? Does stay-at-home policy work? How should economy be opened? These are big questions and answers to them literally altered lives of hundreds of millions of people. We humbly pushed forward, scrutinizing every step. Ultimately, what we claimed still stands today; our findings have not been disproven.

Applying innovation to a business problem, at Kumo.ai, we're revolutionizing how predictive AI is being applied to enterprise data. We thought of relational databases as heterogeneous hypergraphs and by combining them with Graph Neurla Networks we invented a new field of Relational Deep Learning. This is another example of how a shift in perspective or asking different questions led us to discover a new continent. We can now train predictive AI without any feature engineering. It is data-driven end-to-end, which brings state of the art accuracy as well as speed. With Kumo one can quickly build accurate predictive models for important business tasks, such as determining which customer is likely to churn, which product a customer will purchase, or how much of a product should be manufactured to meet demand.

When did you first discover KDD and why is the organization important to you?

My initial involvement with KDD was as an undergraduate student in Slovenia, when a friend and I participated in – and won – the KDD. It was amazing to participate at KDD and meet great researchers. I've remained actively engaged with KDD ever since then. KDD is incredibly unique. KDD sits at the crossroads, where two rivers come together: an open-ended academic river and a high-impact industry river. At this convergence, the nutrients of both mixes, creating excitement and opportunity seldom found in other professional organizations. At its core, science is highly collaborative, and KDD offers an ideal community for exchange of ideas and ongoing conversation with similarly minded innovators.

What advice do you have for data scientists at the beginning of their careers?

We've already touched on the importance of zeroing in on the right questions to ask. It's crucial to determine the right problem to work on, and the right time to tackle it. Rather than being overly confident in your own conviction, I recommend remaining humble and being receptive to what the world – and the data - can teach you. AI, data science, and machine learning are all about iterative progress, so data scientists must be open to learn at every step, ready to admit when they are wrong, and willing to adjust course if that's where the data leads.

Professor Leskovec, thank you for taking the time to speak with us and share your perspective. Again, congratulations on receiving the ACM SIGKDD 2023 Innovation Award.

Marginal Nodes Matter: Towards Structure Fairness in Graphs

Xiaotian Han[1] Kaixiong Zhou[2] Ting-Hsiang Wang[1] Jundong Li[3] Fei Wang[4] Na Zou[1]
[1]Texas A&M University, [2]Rice University, [3]University of Virginia, [4]Weill Cornell Medicine
{han,thwang1231,nzou1}@tamu.edu, jundong@virginia.edu, few2001@med.cornell.edu

ABSTRACT

In social network, a person located at the periphery region (marginal node) is likely to be treated unfairly when compared with the persons at the center. While existing fairness works on graphs mainly focus on protecting sensitive attributes (e.g., age and gender), the fairness incurred by the graph structure should also be given attention. On the other hand, the information aggregation mechanism of graph neural networks amplifies such structure unfairness, as marginal nodes are often far away from other nodes. In this paper, we focus on novel fairness incurred by the graph structure on graph neural networks, named *structure fairness*. Specifically, we first analyzed multiple graphs and observed that marginal nodes in graphs have a worse performance of downstream tasks than others in graph neural networks. Motivated by the observation, we propose **S**tructural **Fair** **G**raph **N**eural **N**etwork (`SFairGNN`), which combines neighborhood expansion based structure debiasing with hop-aware attentive information aggregation to achieve structure fairness. Our experiments show `SFairGNN` can significantly improve structure fairness while maintaining overall performance in the downstream tasks.

1. INTRODUCTION

Recent years have witnessed a surge of research interests in graph machine learning for different applications, such as social networks [26, 55], protein-protein interaction networks [3], knowledge graphs [20,45,73], and Spatio-Temporal forecasting [14]. Among them, graph neural networks (GNNs) [63,64,72,74], as a family of deep learning based algorithms, have become an essential paradigm for graph analysis due to their superior modeling performance. Despite their success, it has been shown that these algorithms often suffer from fairness issues for decision-making [16, 18, 23, 24, 42, 48, 61, 70]. To mitigate this issue, a vast majority of existing works on graph fairness [11, 24, 42, 48, 54] focus on obtaining fair node representations that are invariant to the change of the protected attributes such as age, gender, and ethnicity.

Complementing existing efforts on learning fair representations regarding protected attributes [11, 24, 42, 48, 54], we investigate a novel problem on the fairness issue of GNNs from the *structure* perspective. In particular, graph structure fairness should be paid more attention to because of the following key reasons: 1) Marginal nodes in the graph should not be automatically depreciated by graph learning algorithms. For instance, in the Twitter network, a tweet posted by a user with few followers may receive limited attention, although the tweet could be insightful and valuable for the whole community. As another example, in a job market network such as the LinkedIn network, people should not be neglected just because they are at the periphery of the social network, a situation commonly faced by individuals from underprivileged communities. 2) As one of the most popular methods in graph analysis, GNNs exacerbate structure unfairness because their built-in information aggregation mechanism exploits the egocentric structure of nodes. In other words, each node only receives information from nearby nodes in information aggregation.

Despite its importance and necessity, understanding and achieving structure fairness on GNNs is non-trivial due to the following two challenges. First, a proper measurement of structure fairness is missing, although a quantitative measurement is essential to detect and compensate for *structurally* underprivileged nodes. Second, given that GNNs use an information aggregation mechanism to update node representation from neighbors, but the information aggregation mechanism implicitly causes unfair decision-making, it remains a challenging problem to design an effective graph neural network framework to mitigate the unfairness problem of information aggregation.

To overcome the challenges mentioned above, we first validate that existing graph neural networks suffer from structure fairness problems and then propose to address them accordingly. We first conduct an experiment to validate that structure fairness is highly correlated to the performance of downstream tasks in existing GNNs. Since the information aggregation mechanism of GNNs essentially enriches the node information by aggregating information from their neighbors, centrality [10,43] is an intuitive indicator of graph structure, which contains the information between nodes and their neighbors. To this end, we adopt two centrality measurements (closeness centrality [36] and eigenvector centrality [9]) to identify the marginal nodes.

To address structure fairness in graph neural networks, we propose a simple-yet-effective **S**tructural **Fair** **G**raph **N**eural **N**etwork (`SFairGNN`), which mitigates the structure unfairness issue of GNNs without sacrificing their performance on downstream tasks. Specifically, `SFairGNN` has two essential components: 1) neighborhood expansion and 2) hop-aware attentive information aggregation. Neighborhood expansion compensates marginal nodes by introducing new neighbors to them during the training phase. On the other hand, hop-aware attentive information aggregation optimizes how

the new information should be properly and effectively integrated into each node based on downstream tasks. Our investigation and proposed model will help researchers and society understand how to identify and mitigate structure unfairness in graph-based decision-making. In summary, the **contributions** of this work are as follows:

- We conducted an analysis of the performance of downstream tasks on marginal nodes and found that they were significantly disadvantaged, leading to unfair decision-making. Our investigation led us to the initial observation that the graph structure was responsible for the unfairness suffered by these nodes.
- To address the issue of unfairness caused by the graph structure, we proposed a new graph neural network model called `SFairGNN`. Our proposed model is specifically designed to mitigate the impact of structural unfairness in graph neural networks.
- To evaluate the performance of `SFairGNN`, we conducted experiments on real-world datasets. Our results showed that `SFairGNN` outperformed state-of-the-art GNN models in terms of fairness, without sacrificing performance. Our experiments demonstrate the effectiveness of `SFairGNN` in reducing unfairness and improving overall performance in graph-based machine learning models.

The rest of the paper is organized as follows. In Section 3, we provide an overview of the necessary background for `SFairGNN`, which includes an introduction to graph neural networks and metrics for measuring the structure of graphs. Section 3 delves into the issue of fairness in graph structures. Next, in Section 4, we present our proposed fairness-aware graph neural network, `SFairGNN`. We discuss the experimental findings and analysis in Section 5. Finally, in Sections 6 and 7, we discuss the related work and provide concluding remarks and suggestions for future research.

2. PRELIMINARIES

In this section, we introduce the essential preliminaries for our proposed methods, including graph neural networks, as well as two centrality metrics of a graph: closeness centrality and eigenvector centrality.

2.1 Graph Neural Networks

Most graph neural networks (GNNs) [30, 39, 59, 62] adopt a message-passing mechanism to update node representations over the graph iteratively. Without loss of generality, we present a general framework of GNNs, which can be divided into two steps: aggregation and transformation. Generally, at the k-th layer in GNNs, the hidden representation $\mathbf{x}_i^{(k)}$ of node n_i is updated as follows:

$$\mathbf{h}_i^{(k)} = \text{AGGREGATE}(\{a_{ij}^{(k)}\mathbf{W}^{(k)}\mathbf{x}_j^{(k-1)} : j \in \mathcal{N}_i\}),$$
$$\mathbf{x}_i^{(k)} = \text{ACT}(\text{COMBINE}(\mathbf{x}_i^{(k-1)}, \mathbf{h}_i^{(k)})),$$

where $\mathbf{W}^{(k)}$ denotes the trainable weight, and $a_{ij}^{(k)}$ denotes the edge weight between nodes i and j, and $\mathcal{N}_i$ denotes the set of neighbors adjacent to node i. AGGREGATE$(\cdot)$ is an aggregation function, which is applied to aggregate the node representations of all neighbors. Finally, COMBINE$(\cdot)$ and ACT$(\cdot)$ are applied to combine neighbor information and activate node representations, respectively.

2.2 Indicators of Structure

To account for the impact of graph structure toward fair decision-making in GNNs, we first need to identify the marginal nodes in the graph. To reflect node sociological origin [36] and the graph structure at the node level, we use two important centralities (i.e., closeness centrality and eigenvector centrality) to quantify the structure of each node. Closeness centrality [36] of a node measures its average nearness (inverse distance) to all other nodes in the graph. Eigenvector centrality [9] of a node measures the influence it has on all other nodes in the graph.

Closeness Centrality Closeness centrality [5, 49] of a node is the reciprocal of the sum of the shortest path distances from the node to all other nodes on the graph. The nodes with a high closeness centrality score have the shortest distances to all other nodes, which easily access other nodes and influence other nodes. For instance, in a social network, a person with a lower mean distance from others might find that their opinions reach others more quickly than the opinions of someone with a higher mean distance. Formally, the mathematical expression of closeness centrality of node n_i is:

$$c(n_i) = \frac{N-1}{\Sigma_{n_i} d(n_i, n_j)}, \tag{1}$$

where $d(n_i, n_j)$ is the shortest-path distance between nodes i and j, N is the number of nodes in the graph.

Eigenvector Centrality Eigenvector centrality is a measure of the influence of each node in a graph [9]. A high eigenvector centrality score for a node indicates that it is connected to many other nodes with high scores. The eigenvector centrality score for node i is calculated as the i-th element of the vector $\mathbf{x}$, which is defined by the equation $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, where $\mathbf{A}$ is the adjacency matrix of the graph and λ is the corresponding eigenvalue. According to the Perron-Frobenius theorem [47], if λ is the largest eigenvalue of $\mathbf{A}$, then there exists a unique solution $\mathbf{x}$, all of whose entries are positive. [29] Throughout this paper, we use the indicator variable s_i to represent the centrality metric of node i. Specifically, s_i can take either closeness centrality or eigenvector centrality, depending on the context.

3. PROBLEM FORMULATION

In the following subsections, we introduce and analyze the structure fairness in graph neural networks and propose how to measure the structure fairness.

3.1 Preliminary Experiments

To explore the structure unfairness in GNNs, we perform node classification task with representative GNNs, i.e., graph convolutional networks (GCN) [35] and graph attention networks (GAT) [59]. Specially, we cluster nodes in the graph into multiple bins based on their indicators of structure (i.e., closeness centrality and eigenvector centrality). Then we calculate the average accuracy of each bin and compare the accuracy difference between bins.

Experimental Results. We present the experimental results on `CoraFull` and `PubMed` dataset with closeness and eigenvector centrality in Figure 1. The X-axis represents different levels of closeness/eigenvector centrality, while Y-axis represents the prediction accuracy.

Results Analysis. The lower classification accuracy of marginal nodes shows that they are treated unfairly by GCN

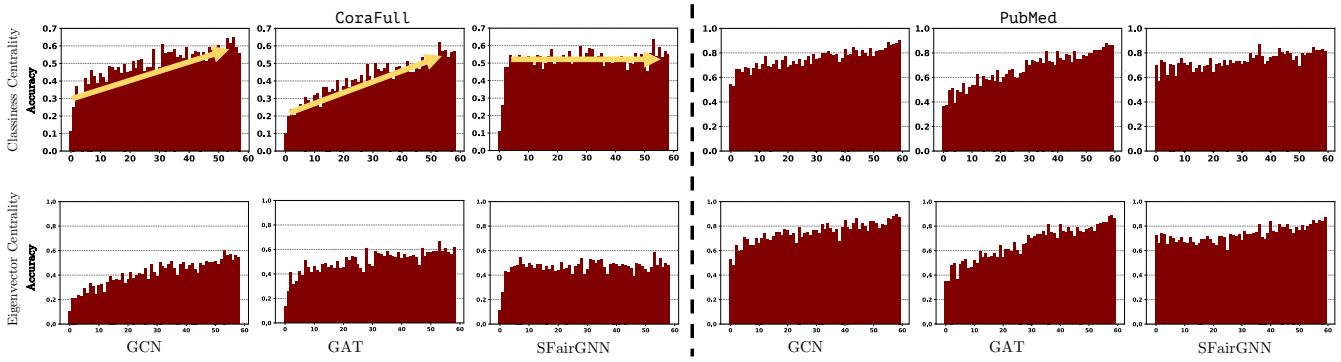


Figure 1: The distribution of node classification accuracy with respect to closeness/eigenvector centrality on `CoraFull` and `PubMed` datasets. The X-axis indicates closeness/eigenvector centrality and Y-axis indicates classification accuracy. We can see that the performance of GCN and GAT is biased towards nodes with higher closeness/eigenvector centrality. In contrast, `SFairGNN` has a more balanced classification performance distribution and implies a lower correlation between downstream task performance and closeness/eigenvector centrality, implying a lower correlation between downstream task performance and centrality scores. ➡ on the top-left figure clearly indicates that GCN and GAT are biased towards nodes with higher closeness/eigenvector centrality, while our method does not exhibit this bias.

and GAT. Therefore, it is clear that structure unfairness in GCN and GAT can lead to discriminative results for marginal individuals, especially when life-changing decisions are at stake. Such unfairness can lead to discriminatory results for marginalized individuals, particularly in cases where significant decisions are at stake. As shown in the figures on the top row, GCN and GAT tend to achieve higher classification accuracy for nodes with higher closeness centrality and lower classification accuracy for nodes with lower closeness centrality. The experimental results presented in this section indicate that `SFairGNN` is capable of achieving structural fairness for downstream tasks.

Similarly, the experiment results on another indicator of structure (i.e., eigenvector centrality) have a similar trend. From the figure, GCN and GAT bias towards nodes with higher eigenvector centrality. In contrast, `SFairGNN` has comparable average node classification accuracy and has more even performance distribution and a lower correlation between performance and eigenvector centrality. The figures in the second row in Figure 1 show that eigenvector centrality is indeed highly correlated to classification accuracy with GCN and GAT models. The observation here is quite similar to the experiment using closeness centrality as an indicator of structure.

3.2 Why GNNs are Structurally Unfair?

Here, we analyze the root cause of the unfair decision-making of GNNs. As we have discussed, the neighborhood aggregation mechanism of GNNs treats nodes with different structures unfairly, especially for the marginal nodes. To further illustrate the unfair neighbor aggregation mechanism, we use Figure 2a as a toy example. There are three groups of nodes with different egocentric structures, which are marked in yellow, blue, and green. The yellow indicates the central nodes, while the green indicates the marginal nodes. And the blue nodes situate between them. Since the green nodes and the blue nodes obtain less information than the yellow nodes, the GNNs will treat these three groups of nodes differently. Through the analysis of the toy example, we conclude that the aggregation mechanism of GNNs can lead to structure unfairness and put the marginal nodes into a disadvantaged situation since it is inherently difficult for marginal nodes to receive the global information of the graph. Although the aggregation mechanism in GNNs improves the performance of the graph analysis, it amplifies the structure fairness.

3.3 Measurement of Structure Fairness

Since centralities and classification accuracy are both continuous variables, we propose the following two metrics to measure structure fairness. One is the correlation between the indicator of graph structure and the probability of correct classification. Another one is the variation of the average classification accuracy for nodes in different bins within different egocentric structure ranges.

Pearson Correlation Coefficient (PCC) is a statistical metric that measures the linear correlation between two variables [6]. In our scenario, we use PCC to assess the linear correlation between closeness centrality and the probability of correct classification. A PCC value in the range of $[-1, 0]/[0, 1]$ indicates that two variables are negatively/positively correlated, respectively. A low PCC value suggests that the algorithm is preserving structural fairness.

Standard Deviation (STD) quantifies the variation of average classification accuracy of nodes in different bins within different ranges of centrality values. Specifically, we cluster nodes into different bins based on their centrality values. We first calculate the mean classification accuracy for each bin and then calculate their STD. A small STD value means that a model has similar classification accuracy for nodes in different bins, which implies that nodes are treated fairly. From Figure 1, the STD of the GCN, GAT, and `SFairGNN` are $0.2480, 0.2935, 0.1583$ for the `CoraFull` dataset, respectively. As our proposed `SFairGNN` has the lowest STD, it beats GNN and GAT in terms of structure fairness based on the closeness/eigenvector centrality.

4. THE PROPOSED METHOD

In this section, we introduce **S**tructural **Fair** **G**raph **N**eural **N**etwork (`SFairGNN`), whose fundamental intuition is to allow marginal nodes to better obtain the information from

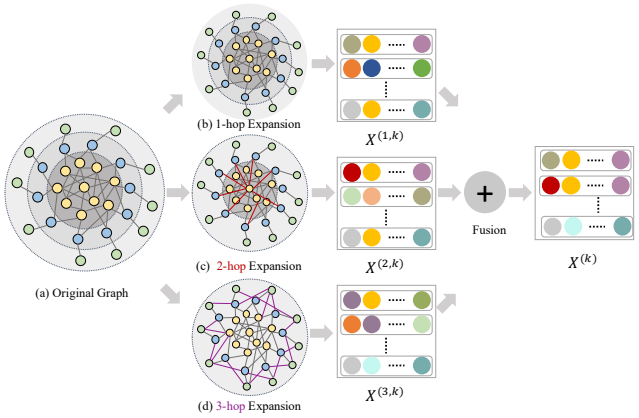


Figure 2: An illustration of our proposed method, specifically an example of the 2-layer SFairGNN. After the neighborhood expansion and hop-wise information aggregation, we obtain the embedding matrices for these three hops: $\{\mathbf{X}^{(1,k)}, \mathbf{X}^{(2,k)}, \mathbf{X}^{(3,k)}\}$. They are combined with a fusion function to output the final embedding $\mathbf{X}^{(k)}$ at k-th graph convolution layer.

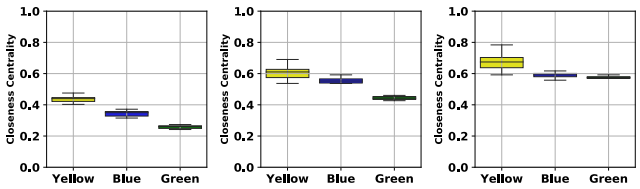


Figure 3: The distribution of closeness centrality for different-hop neighbors. The X-axis represents nodes colored yellow, blue, and green, while the Y-axis denotes closeness centrality. In (a), we have the original graph, while (b) and (c) show the graphs after 2-hop and 3-hop neighborhood expansions, respectively. We observe that the distribution of closeness centrality becomes more stable as the hop number h increases.

the center of the graph. The framework of SFairGNN shown in Figure 2 is composed of two essential components: 1) the neighborhood expansion component that creates new neighbors for marginal nodes in the graph and brings them closer to the center of the graph; 2) the hop-aware attentive information aggregation component that leverages attention mechanism to fuse representations of neighbors at different hops towards a fair embedding for all nodes. In the following sections, we will introduce the details of neighborhood expansion and hop-aware attentive information aggregation. To illustrate the proposed SFairGNN, we use closeness centrality as an example indicator of graph structure.

4.1 Neighborhood Expansion Based Structure Debiasing

Before delving into the neighborhood expansion based structure debiasing, we first investigate the distribution of closeness centrality over a synthetic graph to better understand the unfairness of the traditional information aggregation of GNNs. We plot the distribution of closeness centrality in Figure 3a based on the graph shown in Figure 2a. The synthetic graph in Figure 2a contains central, middle, and marginal nodes colored yellow, blue, and green. It is observed from Figure 3a that the closeness centrality varies significantly among the three colors, where the marginal nodes are much lower than the central nodes.

Motivated by the above observation, we propose the neighborhood expansion method to debias the structure unfairness by strengthening the marginal nodes' connections to the center of the graph. Specifically, a heuristic threshold is adopted to determine whether a node is marginal or not. We name this threshold as margin line and denote it as *line*. The margin line serves as an important hyperparameter of the proposed method. The nodes with closeness centrality $s_i \leq line$ are regarded as marginal nodes; otherwise, they are central nodes. For each node n_i, we expand its original neighbor set $\mathcal{N}_i$ to a hyperset $\{\mathcal{N}_i^{(1)}, \mathcal{N}_i^{(2)}, \cdots, \mathcal{N}_i^{(h)}\}$ to indicate neighbors within h hops. The 1-hop neighbor set $\mathcal{N}_i^{(1)}$ is given by $\mathcal{N}_i$, which is generated based on the original adjacency matrix $\mathbf{A}$. We define a debiased adjacency matrix $\tilde{\mathbf{A}}$ with each element given as:

$$\tilde{\mathbf{A}}_{i,j} = \begin{cases} \mathbf{A}_{i,j}, & \text{if } s_i <= line \text{ or } s_j <= line, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where s_i is the indicator of structure for node n_i. In the debiased matrix $\tilde{\mathbf{A}}$, we only strengthen the connections of marginal nodes. Then we use the h-order power of debiased matrix $\tilde{\mathbf{A}}^h$ to generate neighbor set $\mathcal{N}_i^{(h)}$. We obtain a series of neighbor sets of node n_i from different hops $\{\mathcal{N}_i^{(1)}, \mathcal{N}_i^{(2)}, \cdots, \mathcal{N}_i^{(h)}\}$, which will be used for hop-aware attentive information aggregation.

Figure 2b, Figure 2c, and Figure 2d represent the graphs obtained by neighborhood expansions with $h = 1, 2$ and 3, respectively. Note that the graph with 1-hop expansion is the same as the original graph. To verify the effectiveness of neighborhood expansion, we illustrate the closeness centrality distributions of graphs with different hop expansions in Figure 3. As we can see, increasing hop numbers for neighborhood expansion shrinks the closeness centrality gap between different node groups. The empirical study shows that neighborhood expansion can effectively alleviate the structure unfairness in terms of closeness centrality.

4.2 Hop-Aware Attentive Aggregation

Since the neighbors of different hops contain different amounts of information, we propose hop-aware attentive information aggregation to provide customized information aggregation from neighbors of different hops. The overview is illustrated in Figure 2. Specifically, at the k-th graph convolution layer, let $\mathbf{X}^{(k-1)}$ and $\mathbf{X}^{(k)}$ denote the input and output embedding matrices of all the nodes, respectively. The signal processing at each layer includes two steps: 1) hop-wise information aggregation and 2) hop fusion. With the series of neighbor sets $\{\mathcal{N}_i^{(1)}, \mathcal{N}_i^{(2)}, \cdots, \mathcal{N}_i^{(h)}\}$ obtained by the neighborhood expansion, we can compute the hop-wise embedding matrices $\{\mathbf{X}^{(1,k)}, \mathbf{X}^{(2,k)}, \cdots, \mathbf{X}^{(h,k)}\}$ at the k-th layer. Then we fuse these embedding matrices to obtain the final output node representation $\mathbf{X}^{(k)}$. We present the details of the two steps in the following:

Hop-Wise Information Aggregation Given neighbor sets $\mathcal{N}_i^{(h)}$ of all the nodes, we aim to obtain embedding $\mathbf{X}^{(h,k)}$ at the k-th layer for h-th hop neighbors. Considering the different importance of expanded neighbors, we leverage the attention mechanism [59] to learn the attention coefficients of different hop neighbors automatically. Specifically, given neighbor j for node i, attention coefficients is calculated by:

$$\alpha_{ij}^{(h,k)} = \frac{\exp\left(f\left(\mathbf{W}^{(h,k)} \cdot \left[\mathbf{x}_i^{(k-1)} || \mathbf{x}_j^{(k-1)}\right]\right)\right)}{\sum_{m \in \mathcal{N}_i^{(h)}} \exp\left(f\left(\mathbf{W}^{(h,k)} \cdot \left[\mathbf{x}_i^{(k-1)} || \mathbf{x}_m^{(k-1)}\right]\right)\right)}, \tag{3}$$

where $||$ is the concatenation operation. $f(\cdot)$ is the activation function. $\mathbf{W}^{(h,k)}$ is the trainable weight matrix for the h-th hop neighbors at the k-th layer. $\mathbf{x}_i^{(k-1)}$ is the representation of node i, which is the i-th row of embedding matrix $\mathbf{X}^{(k-1)}$. With the attention coefficient $\alpha_{ij}^{(h,k)}$, the embedding of node n_i at the k-th layer within h-th hop neighbors is given by:

$$\mathbf{x}_i^{(h,k)} = \mathrm{AGG}\left(\{\alpha_{ij}^{(h,k)} \mathbf{W}^{(h,k)} \mathbf{x}_j^{(k-1)} : j \in \mathcal{N}_i^{(h)}\}\right). \tag{4}$$

Then we have $\mathbf{X}^{(h,k)} = [\mathbf{x}_1^{(h,k)}, \mathbf{x}_2^{(h,k)}, \cdots, \mathbf{x}_N^{(h,k)}]^T$, where N is the total number of nodes in the graph.

Hop Fusion After we obtain a series of hop-wise embedding matrices $\{\mathbf{X}^{(1,k)}, \mathbf{X}^{(2,k)}, \cdots, \mathbf{X}^{(h,k)}\}$, we adopt fusion function $\mathcal{F}$ to fuse them into final output embedding:

$$\mathbf{X}^{(k)} = \mathcal{F}\left(\mathbf{X}^{(1,k)}, \mathbf{X}^{(2,k)}, \cdots, \mathbf{X}^{(h,k)}\right). \tag{5}$$

Fusion methods $\mathcal{F}$ could be implemented by different functions, such as average or maximum functions. We perform experiments to compare the effects of different fusion methods [50] (i.e., SEQ, AVG, and MAX) in Section 5.3. Note that in the used datasets, the input node features usually play a deterministic role in deciding the nodes' labels. These informative node features will prohibit us from focusing on the influence of structure on the following classification task. Therefore, instead of taking the node features as input, we randomly initialize $\mathbf{X}^{(0)}$ as input node features and set it trainable during the training phase.

4.3 Discussion

Model Analysis. Here we analyze why the two key components of SFairGNN promote structure fairness: 1) neighborhood expansion strengthens the connection of marginal nodes in the graph, which positions them closer to the center of the graph and thus improves their accessibility to information from other nodes. 2) hop-aware attentive information aggregation enables SFairGNN to learn customized weights for information from neighbors of different hops away, which adaptively considers the varied strength of influence from neighboring nodes.

Avoid Constraining Loss over Fairness SFairGNN promotes the structure fairness of marginal nodes by neighbor expansion instead of explicitly optimizing fairness in loss function for two reasons. First, SFairGNN intends to improve fairness not for the measure's own sake, but for a "healthier" graph structure. This minimizes the trade-off between the overall performance and fairness for SFairGNN, as lowering the performance of "privileged" nodes is the shortest path to achieve fairness for direct optimization. From Table 2, we verify SFairGNN does not sacrifice the model performance for fairness. Second, compared to direct optimization, neighbor expansion is drastically more efficient. This is because computing centrality scores for every training iteration is prohibitively expensive.

Comparison to Related Works. Our proposed method focuses on structure fairness, while the previous works [11,17, 48,57] mainly focus on learning fair node representations regarding protected attributes. For the two methods which resemble SFairGNN the most, we discuss the difference between them and our proposed method below. Fairwalk [48] proposes a fairness-aware embedding method based on the random walk technique for attributed networks. FairGNN [17] uses adversarial debiasing and covariance constraint to regularize the GNN to obtain fair node representations and predictions. Both of them focus on the categorical sensitive attributes, while our proposed method does not require any sensitive features and aims to achieve structure-level debias.

Comparison to Cold-Start Our problem is different from the cold-start problem to a large extent, which does not aim to improve fairness across users but improve the recommendation performance for new users. In addition, while cold-start in recommender systems gradually warms up, unfairness in graphs, especially in the social context, can worsen over time and thus an algorithmic solution is needed.

5. EXPERIMENTS

In this section, we empirically evaluate the effectiveness of SFairGNN in node classification tasks by answering the following questions:

- **Q1:** To what extent can SFairGNN outperform the current GNNs in simultaneously achieving high classification accuracy and maintaining structure fairness?
- **Q3:** What is the impact of using different fusion methods on the performance of SFairGNN?
- **Q2:** How do the hyperparameters of SFairGNN, such as the hop number and margin line, affect its ability to mitigate bias?

5.1 Experimental Setting

In this section, we outlines our experimental setting for evaluating the effectiveness of SFairGNN in mitigating bias and achieving structural fairness. Specifically, we provide a comprehensive overview of the benchmark datasets, the baseline methods, and the implementation details both our proposed method and the baseline models.

Baselines In the experiments, we consider the widely used graph neural networks including graph convolutional networks (GCN) [35], graph attention networks (GAT) [59], and GraphSAGE [30] as baseline methods. The details of baseline methods are listed as follows:

- **GCN** [35] is the pioneer work of graph neural networks. GCN is a scalable approach for semi-supervised learning on graph-structured data that is based on an efficient variant of convolutional neural networks.
- **GAT** [59] incorporates the attention mechanism into the propagation step. This method computes the hidden states of each node by attending to its neighbors following a self-attention strategy.
- **GraphSAGE** [30] efficiently generates node representations for previously unseen data. Instead of training individual representations for each node, GraphSAGE learns

Table 1: The statistics of the datasets.

Dataset	#Nodes	#Edges	#Feature	Density	#Class
Cora	2708	10556	1433	0.0014	7
CiteSeer	3327	9104	3703	0.0008	6
PubMed	19717	88648	500	0.0002	3
CoraFull	19793	130622	8710	0.0003	70
Photo	7650	238162	745	0.0042	8
Physics	34493	495924	8415	0.0004	5

a function that generates representations by sampling and aggregating features from a node's local neighborhood.

Datasets Following the previous works, we use six benchmark datasets to study the graph structure fairness on GNN models. They are Cora, CiteSeer, PubMed, CoraFull, Physics, and Photo [65]. The detailed statistics of these datasets are presented in Table 1.

- **Cora**, **CiteSeer**, and **PubMed** are citation networks [65], where nodes and edges denote papers and citations, respectively. Node features are bay-of-words for papers and node labels indicate the fields of papers.
- **CoraFull** [8] is a well-known citation network that contains labels based on the paper topic. CoraFull contains a network extracted from the entire citation network of Cora, while the Cora dataset is its subset.
- **Physics** [51] is co-authorship graph based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. Nodes are authors and edges indicate co-authorship. Node features represent paper keywords for each author's papers, and class labels indicate the most active fields of study for each author.
- **Photo** [51] is part of the Amazon co-purchase graph, where nodes represent goods, and edges indicate the frequency of two goods being bought together. The node features are bag-of-words encoded product reviews and class labels are given by the product category.

Implementation Details We implement both SFairGNN and the baselines with PyTorch [1] and PyTorch Geometric [27]. We initialize model weights with xavier [28] and adopt the Adam optimizer to minimize the cross-entropy loss. We set the initial learning rate to 0.005. We adopt the randomly initialized and trainable embedding vector for each node over a graph instead of node features provided by the dataset. We use 90/10 as the training/test dataset split.

5.2 Main Results on Structure Fairness

To address research question **Q1**, we conduct experiments to compare the proposed SFairGNN with baseline methods and evaluate their fairness performance using the proposed metrics: PCC (Pearson correlation coefficient) and STD (standard deviation). In this section, we utilize closeness centrality and eigenvector centrality as indicators of the graph structure in the experiments. Next, we present an analysis of the experimental results for both closeness centrality and eigenvector centrality.

Structure Fairness with Closeness Centrality Here, we show how the proposed SFairGNN tackles the fairness issue. As shown in Table 2, we can see that SFairGNN achieves the best performance in terms of PCC among all the baseline methods. The PCC values of SFairGNN are all close to zero, which indicates that the closeness centrality and classification accuracy are nearly independent. In other words, SFairGNN successfully tackles the unfairness issue triggered by the biased node egocentric structure. Furthermore, we remark that SFairGNN has a much smaller STD value than baseline methods on most benchmark datasets. This empirical result indicates that SFairGNN dramatically improves the fairness of classification decisions for nodes with different closeness centrality scores. Note that the traditional GNNs use an unfair neighbor aggregation strategy to update node representations, causing marginal nodes to receive less information. On the contrary, SFairGNN expands the neighbors of marginal nodes to debias the unfair node egocentric structures. Such neighborhood expansion strategy enforces structure fairness over the graph. From Table 2, we observe that the overall classification accuracy of SFairGNN is comparable to the baseline methods. SFairGNN does not decrease the overall performance even though helping the marginal nodes may harm the nodes at the center of the graph. On the other hand, SFairGNN can improve structure fairness since the SFairGNN performs better on both the STD and PCC metrics. Thus our proposed model can alleviate the unfairness issue while maintaining the overall model performance for downstream tasks.

Table 2: Comparison between the proposed SFairGNN and baseline methods on closeness centrality. The best scores of STD and PCC are printed in bold. **Improv**↑ represents the fairness improvement, which is calculated based on the baseline GCN. We run each experiment 5 times and report the means. The hop number is set to 3.

Datasets	Metrics	GCN	GAT	SAGE	SFairGNN	↑
Cora	ACC	76.00	66.49	74.85	77.65	−−
	STD	13.15	12.36	12.63	**10.53**	19.9%
	PCC	10.47	26.50	10.96	**0.76**	92.7%
CiteSeer	ACC	54.03	47.55	54.45	53.67	−−
	STD	15.35	15.67	16.23	**13.07**	14.8%
	PCC	14.50	27.30	14.24	**12.92**	10.9%
PubMed	ACC	77.44	66.66	77.82	78.85	−−
	STD	8.12	12.48	07.05	**5.44**	33.0%
	PCC	13.48	3.27	14.68	**5.98**	55.6%
CoraFull	ACC	49.23	40.53	51.86	50.24	−−
	STD	9.65	10.95	09.37	**7.17**	25.7%
	PCC	24.80	29.35	24.20	**15.83**	36.2%
Physics	ACC	90.64	78.79	90.61	91.12	−−
	STD	11.33	17.02	8.79	**7.65**	32.5%
	PCC	37.09	50.59	29.95	**28.56**	23.0%
Photo	ACC	90.66	89.15	90.81	88.89	−−
	STD	**8.71**	10.16	8.91	8.87	−1.8%
	PCC	20.03	32.82	15.50	**10.19**	48.9%

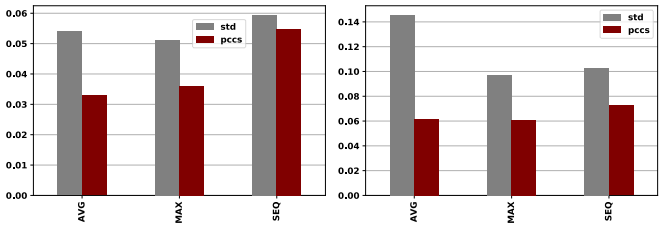
Structure Fairness with Eigenvector Centrality. As shown in Table 3, we can see that SFairGNN achieves the best performance on PCC and STD among the baseline methods. The PCC values of SFairGNN are very low, which indicates the eigenvector centrality and classification accuracy are nearly independent. This observation shows SFairGNN successfully tackles the unfairness issue in the graph. In addition, SFairGNN has a much smaller STD value than the baseline methods, which means SFairGNN dramatically improves the structure fairness for nodes with different eigenvector centrality values.

5.3 Effect of Fusion Methods

To answer the research question **Q2**, we conduct a series of experiments to explore the effect of fusion methods in

Table 3: Comparison between the proposed SFairGNN and baseline methods on eigenvector centrality. The best scores of STD and PCC are printed in bold. **Improv**↑ represents the fairness improvement, which is calculated based on the baseline GCN. We run each experiment 5 times and report the means. The hop number is set to 3.

Datasets	Metrics	GCN	GAT	SAGE	SFairGNN	↑
Cora	ACC	76.41	64.97	75.84	76.00	−−
	STD	9.97	12.52	**9.96**	11.05	−10.49%
	PCC	9.89	14.20	10.37	**0.03**	99.70%
CiteSeer	ACC	53.38	47.61	51.55	53.42	−−
	STD	16.55	16.18	17.47	**11.85**	28.43%
	PCC	16.60	19.69	14.41	**5.96**	64.10%
PubMed	ACC	77.39	65.93	78.44	77.28	−−
	STD	8.24	13.72	7.23	**6.03**	26.63%
	PCC	4.82	6.43	**4.53**	5.04	−4.98%
CoraFull	ACC	50.05	40.64	51.90	47.74	−−
	STD	9.21	10.61	9.18	**6.94**	24.85%
	PCC	9.57	17.34	8.99	**5.43**	43.18%
Physics	ACC	91.14	76.07	90.63	91.74	−−
	STD	11.33	18.20	**8.23**	9.10	19.71%
	PCC	17.98	23.34	12.50	**12.41**	44.25%
Photo	ACC	91.02	88.48	91.09	88.11	−−
	STD	8.11	9.51	8.89	**8.06**	−0.56%
	PCC	8.54	4.79	5.16	**1.64**	80.79%



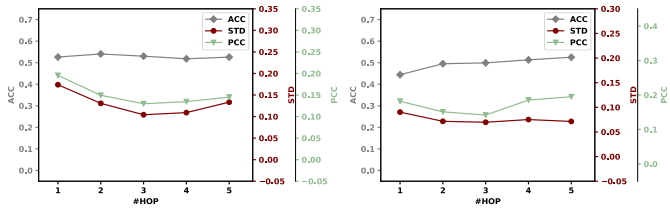
(a) CiteSeer (b) CoraFull

Figure 4: Performance comparison w.r.t. different fusion methods on CiteSeer and CoraFull datasets. We use SEQ, MAX, AVG fusion methods. The experiments show that the fusion method MAX obtains the best performance.

SFairGNN. We consider replacing the fusion method with several different strategies as follows:

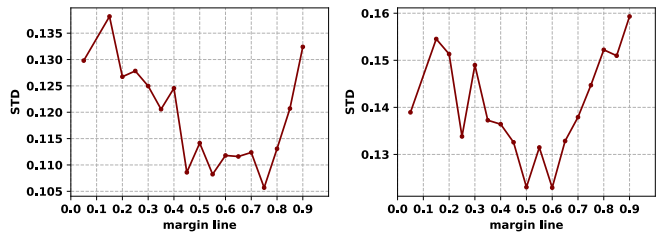
- **SEQ** represents the approach where the model aggregates information of different hop neighbors sequentially. This means the model first aggregates information from first hop neighbors and then moves on to aggregate information from other hop neighbors.
- **MAX** represents the approach where the model selects the maximum value of each element from different hop embeddings for each node. This approach highlights the most important neighbor among different neighbors.
- **AVG** represents the approach where the model takes the average of embeddings of different hop neighbors for each node. This approach treats all the neighbors equally during attentive information aggregation.

One can see from Figure 4 that the fusion method MAX obtains the best performance among the three fusion methods because the MAX fusion method is capable of selecting the most important neighbor automatically to fuse the embeddings. Since the expanded neighbor may introduce noise to the graph structure, the MAX strategy can also reduce the influence of such adverse effects. On the other hand, the SEQ strategy obtains the worst performance on the metric PCC, which may be caused by the fact that SEQ treats different hop neighbors in the same way.



(a) CiteSeer (b) CoraFull

Figure 5: The effect of hop number on the CiteSeer and CoraFull datasets with the closeness centrality. The results show that 1) the values of PCC and STD generally decrease when the hop number is less than 4. 2) the performance drops when the hop number is too high.



(a) CiteSeer (b) CoraFull

Figure 6: The effect of margin line on the CiteSeer and CoraFull datasets. The results show that the value of STD generally decreases and then increases, where the optimal fairness is reached at the middle of the margin line range.

5.4 Hyperparameter Sensitivity Analysis

To answer the research question **Q3**, we conduct experiments to explore the sensitivity of SFairGNN to different hyperparameters, including the hop number h and the margin line *line*. The hop number h indicates the extent of the neighborhood expansion. The margin line *line* determines whether the nodes are marginal nodes or not. For simplicity, we use closeness centrality as the indicator of structure in the experiments for hyperparameter sensitivity analysis.

Effect of Neighbor Hop. The hop number h controls the information reception field of marginal nodes, making it a vital hyperparameter. We conduct experiments on the PubMed and CoraFull datasets considering a series of hop numbers $h \in \{1, \cdots, 5\}$. We consider closeness centrality for brevity. The results are presented in Figure 5. We observed that 1) the values of PCC and STD generally decrease when the hop number is less than or equal to 4. This is because the closeness centrality of marginal nodes is improved with the neighborhood expansion, reducing the variation of the nodes' closeness centrality over the graph.
2) when the hop number is too high, the performance drops because the original graph structure is damaged—being much different from the graph after neighborhood expansion.

Effect of Margin Line. The margin line *line* decides whether a node is marginal or not. To explore its effect, we conduct experiments on the `Cora` and `CiteSeer` datasets with a series of margin line value $line \in \{0.1, \cdots, 0.9\}$ and report the results in Figure 6. One can see from Figure 6 that the value of STD generally decreases and then increases, where the optimal fairness is reached at the middle of the margin line range. This is because when the margin line is too small, the marginal nodes obtain insufficient compensation. On the other hand, the neighborhood expansion incorporates too much noise, which hurts the fairness performance when the margin line is too large. In fact, it should be noted that the margin line has significant sociological meanings, so it should be set seriously for specific fairness issues.

6. RELATED WORKS

This section summarizes two categories of related works, including graph neural networks and fairness in graphs.

6.1 Graph Neural Networks

Graph neural networks (GNNs) [31, 52, 56, 58, 64, 69, 71, 72, 74] have been the new state-of-the-art method to analyze graph-structured data, which are widely applied to social networks [32], and academic citation networks [35], knowledge graphs [21, 73], and to name a few. Starting with the success of graph convolutional network in the semi-supervised node classification task, a wide variety of GNN variants have enhanced and improved the node representation learning and downstream learning tasks [30, 59, 62]. Most of GNNs follow a message-passing strategy to learn node representations over a graph. GNNs apply a neighbor aggregator to update node representations iteratively via combining representations of neighbors and that of the node itself. GraphSAGE [30] learns representations by sampling and aggregating neighbor nodes, whereas GAT [59] uses the attention mechanism to aggregate representations from all neighbors. Besides, there is another branch of GNNs, spectral GCNs [12, 19, 35], which use spectral filters over graph laplacian to define the convolution operation.

6.2 Fairness in Graphs

Fairness [13, 44, 46] in machine learning algorithms attracts a lot of interest from both academic and industrial communities since life-changing decision-making needs to be fair in many sensitive environments, such as loan applications, health care [2], and hiring [4, 7]. One mainstream approach to achieve fairness is to use the in-processing approaches, which reduce the bias in the training data. In-processing approaches typically incorporate one or more fairness constraints into the model optimization process to maximize the performance of downstream tasks and improve fairness simultaneously [25, 67, 68]. Recently, a few works [11, 17, 22, 33, 34, 37, 38, 41, 48] focus on learning fair node representation regarding protected attributes (e.g., gender, age, race). [11] proposes an adversarial framework to enforce fairness constraints on graph learning, which can flexibly accommodate different combinations of fairness constraints during inference. [48] aims to study potential bias issues inherent within graph embedding and propose a fairness-aware embedding method named Fairwalk. FairGNN [17] uses adversarial debiasing and covariance constraint to regularize the GNN to yield fair predictions. There is also a line of works focusing on the fairness in graph structure [15, 40, 53, 60]. The position-aware graph neural network [66] also investigates the "position" of the node in graphs, which is related to our proposed method. A more comprehensive review of related works of fairness in graphs can be found in [23].

7. CONCLUSION

In this work, we propose an `SFairGNN` model that is resilient to structure unfairness in graphs. We show that marginal nodes in the graph have lower prediction performance in existing GNN models, which indicates the existing GNNs suffer from the structure fairness issue. To this end, `SFairGNN` is proposed to preserve fairness without sacrificing the prediction performance of GNNs. Specifically, `SFairGNN` relies on a neighborhood expansion strategy and a hop-aware attentive information aggregation strategy to optimally introduce new information to structurally underprivileged nodes, i.e., marginal nodes. Moreover, we conduct extensive experiments on several graph datasets and validate that our `SFairGNN` can achieve fairer node classification performance than the existing GNNs. In the future, we will consider both structure fairness and node attribute fairness for GNNs since the fairness may be caused by the interactions of structure and attributes. Another promising extension of this work is to study the interpretability of structure fairness.

8. ACKNOWLEDGEMENTS

We would like to thank all the anonymous reviewers for their valuable suggestions. Na Zou is supported by NSF IIS-1900990, IIS-1939716, IIS-2239257. Fei Wang is supported by NSF 1750326 and 2212175 for this research.

9. REFERENCES

[1] P. Adam, G. Sam, C. Soumith, C. Gregory, Y. Edward, D. Zachary, L. Zeming, D. Alban, A. Luca, and L. Adam. Automatic differentiation in pytorch. In *NeurIPS*, 2017.

[2] M. A. Ahmad, A. Patel, C. Eckert, V. Kumar, and A. Teredesai. Fairness in machine learning for healthcare. In *KDD*, 2020.

[3] A. Airola, S. Pyysalo, J. Björne, T. Pahikkala, F. Ginter, and T. Salakoski. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC bioinformatics*, 2008.

[4] G. S. Alder and J. Gilbert. Achieving ethics and fairness in hiring: Going beyond the law. *Journal of Business Ethics*, 2006.

[5] A. Bavelas. Communication patterns in task-oriented groups. *The journal of the acoustical society of America*, 22(6):725–730, 1950.

[6] J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[7] M. Bogen and A. Rieke. Help wanted: An examination of hiring algorithms, equity, and bias, 2018.

[8] A. Bojchevski and S. Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

[9] P. Bonacich. Some unique properties of eigenvector centrality. *Social networks*, 29(4):555–564, 2007.

[10] S. P. Borgatti. Centrality and network flow. *Social networks*, 27(1):55–71, 2005.

[11] A. Bose and W. Hamilton. Compositional fairness constraints for graph embeddings. In *ICML*, pages 715–724. PMLR, 2019.

[12] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

[13] S. Caton and C. Haas. Fairness in machine learning: A survey. *arXiv preprint arXiv:2010.04053*, 2020.

[14] X. Chen, M. Hou, T. Tang, A. Kaur, and F. Xia. Digital twin mobility profiling: A spatio-temporal graph learning approach. In *2021 IEEE 23rd Int Conf on High Performance Computing & Communications*, pages 1178–1187, 2021.

[15] Z. Chen, T. Xiao, and K. Kuang. Ba-gnn: On learning bias-aware graph neural network. In *ICDE*, pages 3012–3024. IEEE, 2022.

[16] M. Choudhary, C. Laclau, and C. Largeron. A survey on fairness for machine learning on graphs. *arXiv preprint arXiv:2205.05396*, 2022.

[17] E. Dai and S. Wang. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *WSDM*, 2021.

[18] E. Dai, T. Zhao, H. Zhu, J. Xu, Z. Guo, H. Liu, J. Tang, and S. Wang. A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. *arXiv preprint arXiv:2204.08570*, 2022.

[19] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 2016.

[20] J. Dong, Q. Zhang, X. Huang, K. Duan, Q. Tan, and Z. Jiang. Hierarchy-aware multi-hop question answering over knowledge graphs. In *International World Wide Web Conference*, pages 2519–2527, 2023.

[21] J. Dong, Q. Zhang, X. Huang, Q. Tan, D. Zha, and Z. Zihao. Active ensemble learning for knowledge graph error detection. In *ACM International Conference on Web Search and Data Mining*, pages 877–885, 2023.

[22] Y. Dong, N. Liu, B. Jalaian, and J. Li. Edits: Modeling and mitigating data bias for graph neural networks. In *Proceedings of the ACM Web Conference 2022*, pages 1259–1269, 2022.

[23] Y. Dong, J. Ma, C. Chen, and J. Li. Fairness in graph mining: A survey. *arXiv preprint arXiv:2204.09888*, 2022.

[24] X. Du, Y. Pei, W. Duivesteijn, and M. Pechenizkiy. Fairness in network representation by latent structural heterogeneity in observational data. In *AAAI*, 2020.

[25] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[26] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin. Graph neural networks for social recommendation. In *The world wide web conference*, 2019.

[27] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[28] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.

[29] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[30] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[31] X. Han, Z. Jiang, N. Liu, Q. Song, J. Li, and X. Hu. Geometric graph representation learning via maximizing rate reduction. In *Proceedings of the ACM Web Conference 2022*, pages 1226–1237, 2022.

[32] X. Huang, Q. Song, Y. Li, and X. Hu. Graph recurrent networks with attributed random walks. In *KDD*, 2019.

[33] Z. Jiang, X. Han, C. Fan, Z. Liu, X. Huang, N. Zou, A. Mostafavi, and X. Hu. Topology matters in fair graph learning: a theoretical pilot study. 2022.

[34] Z. Jiang, X. Han, C. Fan, Z. Liu, N. Zou, A. Mostafavi, and X. Hu. Fmp: Toward fair graph message passing against topology bias. *arXiv preprint arXiv:2202.04187*, 2022.

[35] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[36] D. Koschützki, K. A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski. Centrality indices. In *Network analysis*, pages 16–61. Springer, 2005.

[37] O. D. Kose and Y. Shen. Fair node representation learning via adaptive data augmentation. *arXiv preprint arXiv:2201.08549*, 2022.

[38] P. Li, Y. Wang, H. Zhao, P. Hong, and H. Liu. On dyadic fairness: Exploring and mitigating bias in graph connections. In *International Conference on Learning Representations*, 2021.

[39] H. Ling, Z. Jiang, M. Liu, S. Ji, and N. Zou. Graph mixup with soft alignments. *arXiv preprint arXiv:2306.06788*, 2023.

[40] Z. Liu, T.-K. Nguyen, and Y. Fang. On generalized degree fairness in graph neural networks. *arXiv preprint arXiv:2302.03881*, 2023.

[41] J. Ma, J. Deng, and Q. Mei. Subgroup generalization and fairness of graph neural networks. *NeurIPS*, 34:1048–1061, 2021.

[42] J. Ma, R. Guo, M. Wan, L. Yang, A. Zhang, and J. Li. Learning fair node representations with graph counterfactual fairness. In *WSDM*, pages 695–703, 2022.

[43] P. V. Marsden. Egocentric and sociocentric measures of network centrality. *Social networks*, 2002.

[44] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*, 2019.

[45] D. Q. Nguyen, V. Tong, D. Phung, and D. Q. Nguyen. Node co-occurrence based graph neural networks for knowledge graph link prediction. In *WSDM*, 2022.

[46] D. Pessach and E. Shmueli. Algorithmic fairness. *arXiv preprint arXiv:2001.09784*, 2020.

[47] S. U. Pillai, T. Suel, and S. Cha. The perron-frobenius theorem: some of its applications. *IEEE Signal Processing Magazine*, 22(2):62–75, 2005.

[48] T. Rahman, B. Surma, M. Backes, and Y. Zhang. Fairwalk: towards fair graph embedding. In *IJCAI*, 2019.

[49] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

[50] D. Scherer, A. Müller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[51] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[52] Y. Shi, Y. Dong, Q. Tan, J. Li, and N. Liu. Gigamae: Generalizable graph masked autoencoder via collaborative latent space reconstruction. In *ACM CIKM*, 2023.

[53] H. Shomer, W. Jin, W. Wang, and J. Tang. Toward degree bias in embedding-based knowledge graph completion. *arXiv preprint arXiv:2302.05044*, 2023.

[54] W. Song, Y. Dong, N. Liu, and J. Li. Guide: Group equality informed individual fairness in graph neural networks. In *SIGKDD*, pages 1625–1634, 2022.

[55] Q. Tan, N. Liu, and X. Hu. Deep representation learning for social network analysis. *Frontiers in Big Data*, 2:2, 2019.

[56] Q. Tan, X. Zhang, N. Liu, D. Zha, L. Li, R. Chen, S.-H. Choi, and X. Hu. Bring your own view: Graph neural networks for link prediction with personalized subgraph selection. In *ACM International Conference on Web Search and Data Mining*, pages 625–633, 2023.

[57] X. Tang, H. Yao, Y. Sun, Y. Wang, J. Tang, C. Aggarwal, P. Mitra, and S. Wang. Investigating and mitigating degree-related biases in graph convoltuional networks. In *CIKM*, 2020.

[58] P. Veličković. Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538, 2023.

[59] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[60] R. Wang, X. Wang, C. Shi, and L. Song. Uncovering the structural fairness in graph contrastive learning. *arXiv preprint arXiv:2210.03011*, 2022.

[61] Y. Wang, Y. Zhao, Y. Dong, H. Chen, J. Li, and T. Derr. Improving fairness in graph neural networks via mitigating sensitive attribute leakage. In *ACM SIGKDD*, 2022.

[62] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

[63] L. Wu, P. Cui, J. Pei, L. Zhao, and L. Song. Graph neural networks. In *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 27–37. Springer, 2022.

[64] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE TNNLS*, 2020.

[65] Z. Yang, W. W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

[66] J. You, R. Ying, and J. Leskovec. Position-aware graph neural networks. In *International conference on machine learning*, pages 7134–7143. PMLR, 2019.

[67] M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi. Fairness constraints: Mechanisms for fair classification. *arXiv preprint arXiv:1507.05259*, 2015.

[68] B. H. Zhang, B. Lemoine, and M. Mitchell. Mitigating unwanted biases with adversarial learning. In *AIES*, 2018.

[69] C. Zhang, C. Huang, Y. Li, X. Zhang, Y. Ye, and C. Zhang. Look twice as much as you say: Scene graph contrastive learning for self-supervised image caption generation. In *CIKM*, 2022.

[70] W. Zhang, J. C. Weiss, S. Zhou, and T. Walsh. Fairness amidst non-iid graph data: A literature review. *arXiv preprint arXiv:2202.07170*, 2022.

[71] X. Zhang, Q. Tan, X. Huang, and B. Li. Graph contrastive learning with personalized augmentation. *arXiv preprint arXiv:2209.06560*, 2022.

[72] Z. Zhang, P. Cui, and W. Zhu. Deep learning on graphs: A survey. *IEEE TKDE*, 2020.

[73] Z. Zhang, F. Zhuang, H. Zhu, Z. Shi, H. Xiong, and Q. He. Relational graph neural network with hierarchical attention for knowledge graph completion. In *AAAI*, 2020.

[74] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

Fighting Fire with Fire: Can ChatGPT Detect AI-generated Text?

Amrita Bhattacharjee
School of Computing and AI
Arizona State University
abhatt43@asu.edu

Huan Liu
School of Computing and AI
Arizona State University
huanliu@asu.edu

ABSTRACT

Large language models (LLMs) such as ChatGPT are increasingly being used for various use cases, including text content generation at scale. Although detection methods for such AI-generated text exist already, we investigate ChatGPT's performance as a detector on such AI-generated text, inspired by works that use ChatGPT as a data labeler or annotator. We evaluate the zero-shot performance of ChatGPT in the task of human-written vs. AI-generated text detection, and perform experiments on publicly available datasets. We empirically investigate if ChatGPT is symmetrically effective in detecting AI-generated or human-written text. Our findings provide insight on how ChatGPT and similar LLMs may be leveraged in automated detection pipelines by simply focusing on solving a specific aspect of the problem and deriving the rest from that solution. All code and data is available at https://github.com/AmritaBh/ChatGPT-as-Detector.

1. INTRODUCTION

Recently there have been incredible advancements in large language models (LLMs) that can generate high quality human-like text, with capabilities of assisting humans on a variety of tasks as well. Larger and more expressive models are released to the public frequently, either as public-facing APIs with no access to the model parameters (such as OpenAI's ChatGPT or GPT3.5 family of models [27, 6]) or often with fully open source access (such as LLaMA [36]). Alongside the numerous ways in which these LLMs can aid a human user, act as an assistant and thereby improve productivity, these models can also be misused by actors with malicious intent. For example, malicious actors may use LLMs to generate misinformation and misleading content at scale and publish such content online [32], create fake websites for ad revenue fraud [30], etc. Apart from these malicious use cases, inexperienced users may overestimate the capabilities of these LLMs. The fact that ChatGPT can confidently spew factually incorrect information, yet be fluent and cohesive in the syntax and grammar [2], can fool newer users and mislead them. Users may use ChatGPT to write essays or reports, expecting factuality but then be penalized when flaws are evident. Especially problematic is when people use these models for high-stakes tasks, without realizing the shortcomings of these models, and eventually face dire consequences [5]. Given the accessibility and ease of use of such models, more and more people are using these models in their daily life, perhaps without realizing the nature of the text that is generated, often mistaking fluency and confidence as truthfulness. Therefore, in this work, we focus on the task of distinguishing between human-written and AI-generated text.

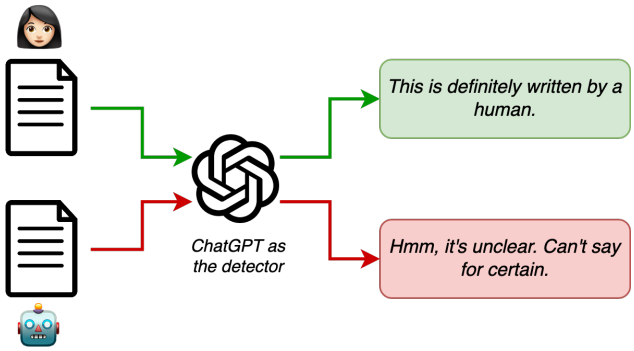


Figure 1: We use OpenAI's ChatGPT as a detector to distinguish between human-written and AI-generated text.

Detection of AI-generated text is very challenging [18], with recent work [31] demonstrating that this challenge is only going to get exacerbated with newer, larger, more capable LLMs. There are existing detection methods, such as feature-based classifiers [20], methods that differentiate human and AI text using statistical measures [14; 26], along with methods that use fine-tuned language models to classify text as human or AI [38; 33]. In this work, however, we want to investigate the capability of ChatGPT (GPT3.5, and the more advanced GPT-4) to differentiate between human and AI-generated text. ChatGPT has been shown to perform well on a variety of NLP and NLU tasks [28], with newer versions of the models being able to use human prompts better than older ones. In order to probe the performance of ChatGPT on detecting AI-generated text, we propose to investigate the following research question:

Can ChatGPT identify AI-generated text from a variety of generators?

The rest of the paper is organized as follows: Section 2 provides a brief overview of LLMs and how they work, Section 3 elaborates on our experiments and experimental settings. Section 4 presents and discusses our results. Section 5 grounds our work in the context of related work, and finally, Section 6 concludes with a discussion of future directions.

2. BACKGROUND: LARGE LANGUAGE MODELS

Large language models (LLMs) are deep neural networks capable of modeling natural language. Most recent LLMs are built using transformer-based architectures, and trained on massive internet-

scale corpora of data. Broadly, LLMs can be of two types: (i) autoregressive (or causal) language models, and (ii) masked language models. Autoregressive LLMs (such as the GPT family of models) are trained to predict the next word or token, given the previous token sequence in a sentence. Formally, at time step t, the model samples a token from the distribution $p(x_t|x_1, ..., x_{t-1})$, based on some pre-defined sampling strategy, and forms the text sequence one token at a time. Sampling strategies may be greedy, top-k, nucleus sampling [19], etc., with the latter two being used commonly in recent LLMs. Masked language models (such as the BERT family of models [11]) are trained on cloze test tasks. Given an input text sequence, $k\%$ of the tokens are masked using a special [MASK] token, and the LLM is trained to predict the tokens in these masked locations in the sequence. Hence these models are bi-directional, unlike autoregressive LLMs. Due to the bi-directional attention mechanisms in masked language models, these are better at natural language understanding tasks, while autoregressive GPT-style models are good at natural language generation [25].

3. CHATGPT AS AI-TEXT DETECTOR

Recent work have used and evaluated ChatGPT for a variety of natural language tasks, annotations of data, few-shot classification settings, and even reasoning and planning (see Section 5). In this work, we want to investigate whether ChatGPT can be used as a detector to identify AI-generated text. ChatGPT was trained on large amounts of text data and we would want to leverage the summarized information or understanding that ChatGPT possesses to try to identify human-written vs. AI-generated text.

Datasets

We use AI-generated text from the publicly available TuringBench dataset [37], which comprises news article style text from 19 different generators. The full list of these 19 generators are: {GPT-1, GPT-2_small, GPT-2_medium, GPT-2_large, GPT-2_xl, GPT-2_PyTorch, GPT-3, GROVER_base, GROVER_large, GROVER_mega, CTRL, XLM, XLNET_base, XLNET_large, FAIR_wmt19, FAIR_wmt20, TRANSFORMER_XL, PPLM_distil, PPLM_gpt2}. Model sizes for each of these 19 generators in provided in Table 1.

For the human-written articles, we use the human articles from the TuringBench dataset. These are news articles from CNN and The Washington Post.

Experimental Setting

We use ChatGPT (gpt-3.5-turbo model endpoint) with version as of June 13, 2023 and GPT-4 with version as of July 12, 2023 as the detector[1]. We experiment with a variety of prompts (as described in the next section) and finally select an appropriate prompt for classifying each news article using ChatGPT. We set the temperature parameter to 0 to ensure minimal variability in the output since we are dealing with a classification task. For each input article, we process the text output produced by ChatGPT or GPT-4 to flag it as one of the three labels: ['human-written', 'AI-generated', 'unclear']. For all ChatGPT experiments, we use the test split of the datasets, which contain around $2,000$ articles. For experiments with GPT-4 as the detector we use the first 500 samples of this test set, due to rate limit constraints on GPT-4.

[1]In this paper, we use the terms 'ChatGPT' to refer to the GPT-3.5 model

Model	# of Parameters
CTRL	1.6B
FAIR_wmt19	656M
FAIR_wmt20	749M
GPT1	117M
GPT2_small	124M
GPT2_medium	355M
GPT2_large	774M
GPT2_xl	1.5B
GPT2_pytorch	344M
GPT3	175B
GROVER_base	124M
GROVER_large	355M
GROVER_mega	1.5B
PPLM_distil	82M
PPLM_gpt2	124M
Transformer_xl	257M
XLM	550M
XLNet_base	110M
XLNet_large	340M

Table 1: Model sizes for the 19 generators in the TuringBench dataset.

Choice of Prompt

Based on some preliminary experiments, we notice that the response from ChatGPT including 'human', 'AI', or 'uncertain' labels for the input text depends significantly on the prompt used. For ease of experimentation and evaluation, we wanted to constrain ChatGPT's responses by using the following prompt for the input text `passage`:

> Task: Identify whether the given passage is generated by an AI or is human-written. Choose your answer from the given answer choices.
> Answer choices: ["generated by AI", "written by human", "unsure"]
> Passage to identify: `<passage>`

But we observe that with this kind of constrained prompt, ChatGPT gets confused. Not only does it fail to generate answers following the given instruction, but it also misclassifies text that it previously classified properly with a simpler prompt. It also provides incorrect labels for inputs that it was previously unsure of.

Hence we revert back to a simpler prompt used in our preliminary experiments:

> 'Is the following generated by an AI or written by a human: `<text>`.'

where `<text>` is the main body text from a human-written or AI-generated news article from our evaluation datasets.

4. RESULTS AND DISCUSSION

In this section we elaborate our main experimental results with ChatGPT, with GPT-4, along with a discussion of the results, followed by some additional experiments.

ChatGPT as Detector

We show the detection performance of ChatGPT (i.e., GPT-3.5) on text from the 19 generators in Figure 2. We see that for the majority of the generators, ChatGPT can identify AI-generated text less than 50% of the samples, and has a very high number of false negatives,

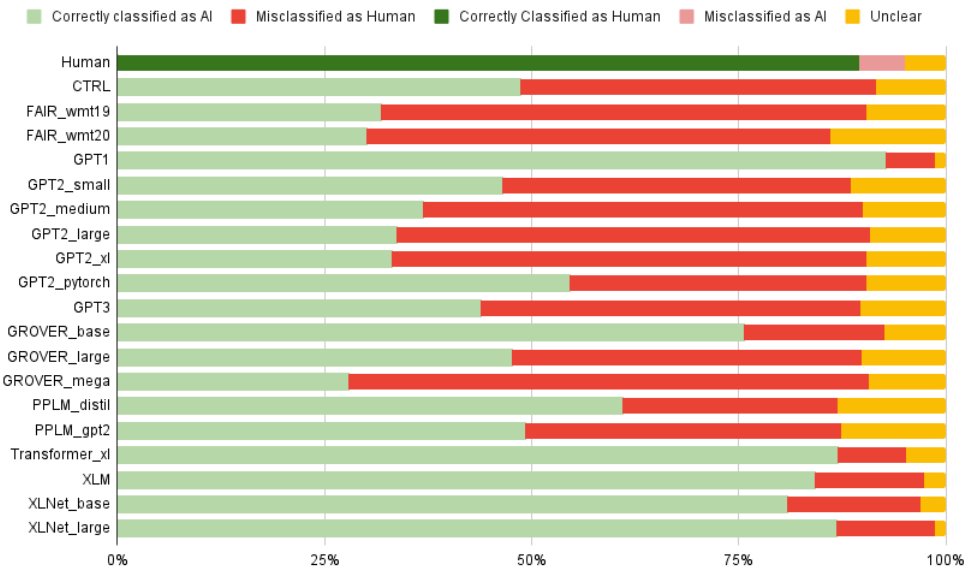


Figure 2: Performance of ChatGPT on texts generated by each of the LLMs in the TuringBench dataset, alongside performance on human-written text from TuringBench (bar at the top).

i.e., AI-generated articles misclassified as human-written. The only satisfactory performance we see is for GPT-1 with around 90% samples correctly identified, and Transformer_xl, GROVER_base, XLM, and the XLNet models with around 75% articles correctly identified as AI-generated. Therefore, our experiment demonstrates that ChatGPT is unable to identify AI-generated text across a variety of generators. We note that even if we flip the label output by ChatGPT and consider that as the final label, i.e. changing all the misclassified 'human-written' labels to 'AI-generated' and the correctly-classified 'AI-generated' labels to now misclassified 'human-written' ones, we do not gain much in performance either, if at all. This is because the proportion of correctly classified and misclassified samples are similar for most of the generators, along with a significant fraction of samples still in the 'uncertain' category.

Next, we investigate ChatGPT's performance on the human-written articles from TuringBench, and we show the distribution of the output labels in Figure 2. Interestingly, we see that for human-written articles, ChatGPT performs significantly better at identifying that the text is human-written, with very few samples being misclassified or even labeled as 'uncertain'.

For the AI-generated texts, we see ChatGPT misclassifies a large fraction of these as human-written. To dig deeper into this phenomenon, we look into how the fraction of false negatives varies with the model size, within a specific model family. Figure 4 shows the fraction of misclassified samples with respect to the different GPT variants with GPT-1 [29] being the smallest and GPT-3 [6] being the largest. We see the percentage of misclassified samples increases with an increase in model size, except for GPT-3, implying that the generation quality becomes more 'human-like' with an increase in the number of model parameters. The discrepancy with GPT-3 having fewer false negatives, even though it is the largest model in our evaluation, seems to be a dataset issue since we see uncharacteristic performance on GPT-3 data even with a fully-supervised classifier. We see a similar trend for the GROVER language model [38] (Figure 5), across three of its size variants: base, large and mega. Similar to the GPT model family, we posit this behavior is due to the text quality becoming better as the model size increases and the models become more expressive. This is also consistent with the performance of other detection methods on these variants of GROVER [38].

GPT-4 as Detector

Similarly, we show the performance of GPT-4 on the 19 generators in Figure 3. We see that for all of the generators, GPT-4 has very good performance, correctly identifying about $97 - 100\%$ of all AI-generated text from the generators. Almost all the text samples are classified as AI-generated, including the human-written texts from TuringBench (top bar in Figure 3). GPT-4 struggles to identify human-written text, and misclassifies over 95% as AI-generated. This would imply that GPT-4 is unable to differentiate between human-written and AI-generated text, for the TuringBench dataset. Interestingly, there are little to no articles (both human and AI-generated) for which GPT-4 outputs an 'unclear' response.

Comparison and Discussion

For the main experiments involving the benchmark TuringBench dataset, we see varied performance between ChatGPT and GPT-4. While there is more *variability* in ChatGPT's performance, GPT-4 tends to label everything as 'AI-generated'. Furthermore, we see a huge difference in the fraction of samples that ChatGPT labeled as 'unclear' vs. the fraction GPT-4 labeled as 'unclear'. This implies that GPT-4 is somehow more confident, even when its predictions are wrong, as in the human-written articles. Hence, these predictions are highly *unreliable*. The degenerate performance of GPT-4 (i.e., labeling everything as one-class, in this case, 'AI-generated') is somewhat unexpected, given the public perception that GPT-4 is better, and more capable than it's previous counterpart ChatGPT. However, recent work has shown empirical evidence that GPT-4's performance might actually be deteriorating over time [8]. This might be due to OpenAI's updates to the GPT-4 model, in order to prevent harmful generations and from people misusing the model.

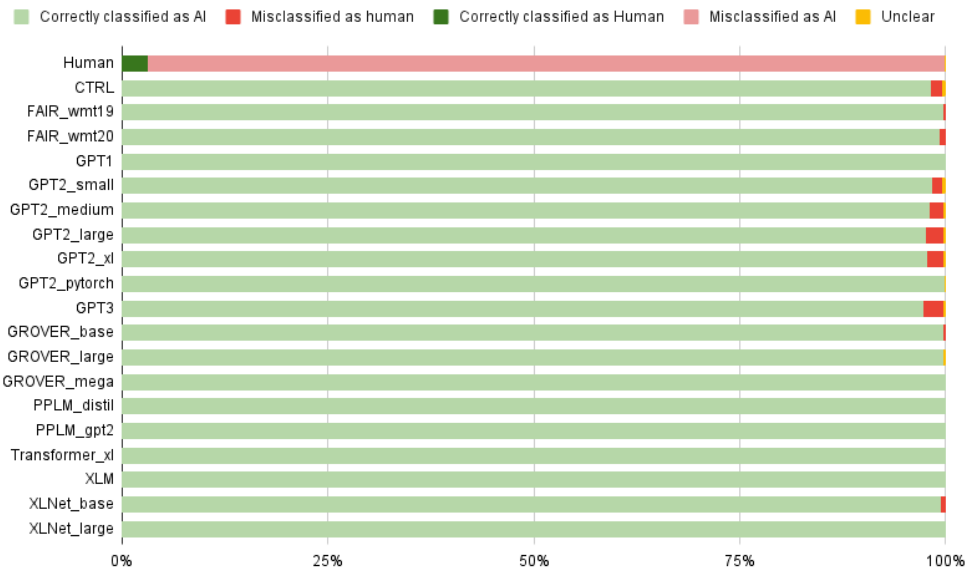


Figure 3: Performance of GPT-4 on texts generated by each of the LLMs in the TuringBench dataset, alongside performance on human-written text from TuringBench (bar at the top).

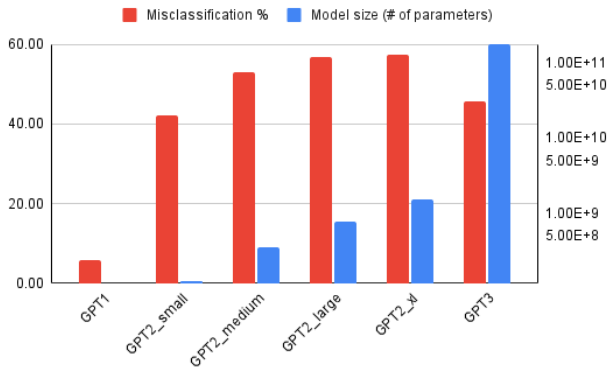


Figure 4: Percentage of AI-generated text misclassified as human-written for different model sizes of GPT. Left-axis: Misclassification %; Right-axis: Model size in log-scale.

Similar to the drift analysis in [8] we used the June 2023 version of ChatGPT and GPT-4 in our experiments, thereby revealing consistent performance degradation of GPT-4, as shown by the authors in [8].

Additional Experiments

Performance on Human-written text from other sources

To test whether ChatGPT and GPT-4's outputs are sensitive to the specific styles, tones, topics, and other features of the human-written text from one specific dataset, we also use the human-written articles from the following datasets:

1. NeuralNews [34]: This dataset consists of human written articles and equivalent articles generated by Grover. For our experiments, we use the human split of the dataset. These are news articles from The New York Times.

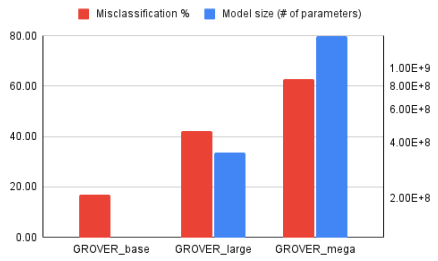


Figure 5: Percentage of AI-generated text misclassified as human-written for different model sizes of GROVER. Left-axis: Misclassification %; Right-axis: Model size in log-scale.

2. IMDb[2]: This dataset comprises of movie reviews, with binary sentiment labels and is originally intended for a sentiment classification task.

3. TweepFake [13]: This dataset comprises Tweets written by humans and also deepfake Tweets. For our purposes, we only use the human-written Tweets.

We show performance of ChatGPT and GPT-4 on these different tyes of human articles in Figure 6 and Figure 7 respectively. For ChatGPT as the detector (Figure 6), majority of human-written text from TuringBench, NeuralNews, IMDb and TweepFake are correctly identified as human-written. There is a significant protion of human-written text labeled as 'unclear' and a much smaller fraction ($\sim 4\%$ and $\sim 6\%$ for TuringBench and IMDb, respectively) misclassified as AI-generated. Hence, we can conclude that the performance of ChatGPT in identifying human-written text is consistent across different sources and styles of human-written text data.

[2]https://huggingface.co/datasets/imdb

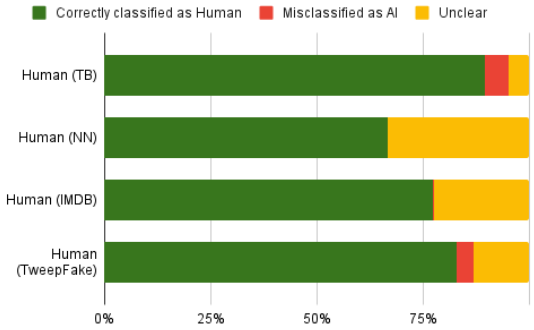


Figure 6: ChatGPT as detector: Distribution of correctly classified and misclassified samples for human-written text from four datasets: {TuringBench, NeuralNews, IMDb, and TweepFake}

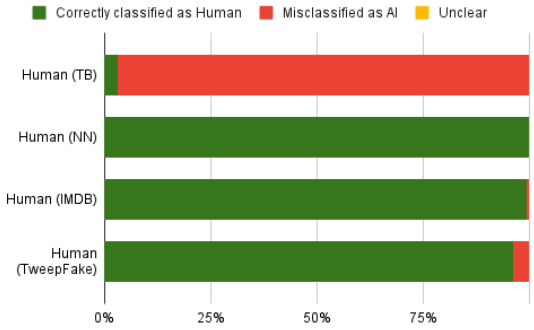


Figure 7: GPT-4 as detector: Distribution of correctly classified and misclassified samples for human-written text from from four datasets: {TuringBench, NeuralNews, IMDb, and TweepFake}

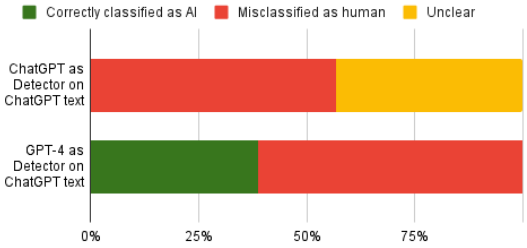


Figure 8: Performance of ChatGPT (top bar) and GPT-4 (bottom bar) on articles generated by ChatGPT.

Interestingly, with GPT-4 as the detector (Figure 7), we see a huge difference in performance across the four human-written datasets. While GPT-4 has almost perfect performance on human texts from NeuralNews, IMDb and very good performance on TweepFake, it misclassifies almost all human text from TuringBench. We think this might be due to dataset specific characteristics: human texts in TuringBench are relatively more 'noisy' than the NeuralNews ones and tend to contain extra characters that might be artifacts from the data collection process. From our experiments, we see that GPT-4 is more sensitive to such dataset artifacts and is therefore unable to classify texts properly. This might imply that GPT-4 cannot be used reliably to identify text written by humans, unlike ChatGPT. This poor performance of recent versions of GPT-4, especially in comparison to ChatGPT, has also been reported in recent work [8], where performance drops of even $\sim 95\%$ have been observed.

Performance on ChatGPT-generated text

Since ChatGPT itself is a language model and hence has the potential of being misused, we are interested in detecting text generated by ChatGPT as well. For this, we use our own ChatGPT-generated data, created in a process similar to the TuringBench dataset [37]. More precisely, we use the same human article sources as in [37], and use the headlines of the articles to generate equivalent ChatGPT articles, using the following prompt:

Generate a news article with the headline '<headline>'.

where <headline> comes from the actual human-written article. For creating this dataset, we use the ChatGPT (GPT-3.5) version as on March 14, 2023. For the rest of this paper, we refer to this dataset as ChatNews, for brevity. We use ChatGPT and GPT-4 as detectors on ChatNews, and report the performance in Figure 8. We see that ChatGPT misclassifies over 50% of ChatNews articles as human-written, and for the remaining, ChatGPT outputs the label 'uncertain'. Only 2 out of 2,000 ChatNews are correctly identified as AI-generated. This poor performance may be due to articles in ChatNews being extremely high quality and human-like, and essentially indistinguishable from actual human-written text. However, when we use GPT-4 as a detector on the same ChatNews dataset, we see more promising results. Interestingly, GPT-4 can correctly identify *some* fraction of ChatNews articles (around 38%) while ChatGPT fails completely. This gives an insight into how newer, larger and perhaps more capable language models may potentially be used to detect text from older language models.

5. RELATED WORK

ChatGPT as a detector or expert

Recent language models ChatGPT and GPT-4 have shown impressive performance on a variety of NLP tasks [28] such as natural language inference, question-answering, sentiment analysis, named entity recognition, etc. There is also empirical evidence of GPT-4 being able to perform discriminative tasks such as identifying PII (personally identifiable information), fact-checking etc. [7]. LLMs have also been evaluated as annotators [12; 17] with recent work showing some LLMs perform at par or even out-perform human crowd workers for text annotation and question-answering [16; 15; 35]. Some recent work also demonstrates how to use an LLM as a controller to use multiple models for AI tasks. Interestingly, there is also evidence [23] that ChatGPT may not perform well for more subjective NLP tasks.

The Landscape of AI-generated text and its detection

The advent of large language models(LLMs) and especially LLM assistants like ChatGPT has normalized the use of AI-generated text for a variety of purposes. Lay persons are also able to use LLMs for work, homework, leisure, or in some cases, even to mislead readers. While such tools can indeed boost productivity and inspire creative thinking, understanding the limitations of these is also important [4; 1; 3]. Given the potential for misuse of LLMs, research on the detection of AI-generated text has gained traction. While several works have shown that humans struggle at identifying AI-generated language [21; 10], a variety of computational methods for detection also exist, including feature-based methods [20; 24], methods exploiting difference in statistical measures across human and AI-generated text [14; 26], more black-box type methods involving fine-tuned language models as the detector backbone [38; 33], etc. With the popularity of ChatGPT and other conversational language models, many commercial AI content detectors have also been released for use, and marketed for use-cases such as plagiarism detection[3][4]. Prominent ones include OpenAI's detector[5], the famous ZeroGPT detector[6], etc. Another recent line of research in the direction of AI-ghenerated text detection is that of watermarking [39; 40; 22; 9] whereby indistinguishable artifacts are embedded into the text, that can be identified by computational or statistical detection methods, but not by a human reader.

6. CONCLUSION & FUTURE WORK

In this work, we investigated the capability of ChatGPT, a large language model, to detect AI-generated text. Our experiments demonstrate an interesting finding that even though ChatGPT struggles to identify AI-generated text, it does perform well on human-written text. This asymmetric performance of ChatGPT can be leveraged to build detectors that focus on identifying human-written text, and thus effectively solve the problem of AI-generated text detection, albeit in an indirect way. A few important takeaways from this empirical analysis would be:

- ChatGPT (GPT3.5) has better, more reliable performance than GPT-4 in identifying AI-generated text vs. human-written text.
- GPT-4 seems to be extremely sensitive to noise and dataset artifacts, such as those that are a result of scraping text from the internet.
- GPT-4's performance is deteriorating over time, and therefore results from using GPT-4 for identifying human-written text may be unreliable and inconclusive.
- The asymmetric performance of ChatGPT may be leveraged in a downstream detection task: ChatGPT (GPT3.5) may be used to specifically identify human-written texts reliably, thereby solving a portion of the AI-generated vs. human-written text detection task.
- Newer, larger generators may be used to detect text from older generators, such as using GPT-4 to identify ChatGPT-generated text.

In future work, we would want to explore *why* this difference in performance exists, and why ChatGPT is much better at identifying human-written text, with significantly less percentage of false negatives (i.e. human-written but misclassified as AI). One hypothesis could be that ChatGPT and these new large language models have been trained on huge corpora of text from the internet. Most of these datasets have data only till 2021, wherein much of the text on the internet was human-written (although with the pervasiveness of ChatGPT and other recent LLMs, the fraction of human to AI text on the internet would possibly change). Therefore, ChatGPT has 'seen' different styles of human writing, how human language flows, and therefore has a better understanding of what human-written text would look like. This subtle capability of ChatGPT may be leveraged to build automated detection pipelines to check the probability of a text being AI-generated. Other interesting future directions for using ChatGPT (or other LLMs) for this task may include few-shot prompting based methods, and ensemble methods leveraging multiple LLMs or feature-based classifiers.

[3]https://docs.thehive.ai/docs/ai-generated-text-detection
[4]https://copyleaks.com/ai-content-detector
[5]https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text
[6]https://www.zerogpt.com/

Acknowledgments

This work is supported by the DARPA SemaFor project (HR001120-C0123) and by the Office of Naval Research via grant no. N00014-21-1-4002. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

7. REFERENCES

[1] H. Alkaissi and S. I. McFarlane. Artificial hallucinations in chatgpt: implications in scientific writing. *Cureus*, 15(2), 2023.

[2] S. Allardice. The confident wrongness of chatgpt, 2023. Accessed on June 29, 2023.

[3] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, et al. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*, 2023.

[4] N. Bian, X. Han, L. Sun, H. Lin, Y. Lu, and B. He. Chatgpt is a knowledgeable but inexperienced solver: An investigation of commonsense problem in large language models. *arXiv preprint arXiv:2303.16421*, 2023.

[5] M. Bohanno. Lawyer used chatgpt in court—and cited fake cases. a judge is considering sanctions, 2023. Accessed on June 29, 2023.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[8] L. Chen, M. Zaharia, and J. Zou. How is chatgpt's behavior changing over time? *arXiv preprint arXiv:2307.09009*, 2023.

[9] M. Christ, S. Gunn, and O. Zamir. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*, 2023.

[10] E. Clark, T. August, S. Serrano, N. Haduong, S. Gururangan, and N. A. Smith. All that's' human'is not gold: Evaluating human evaluation of generated text. *arXiv preprint arXiv:2107.00061*, 2021.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] A. Efrat and O. Levy. The turking test: Can language models understand instructions? *arXiv preprint arXiv:2010.11982*, 2020.

[13] T. Fagni, F. Falchi, M. Gambini, A. Martella, and M. Tesconi. Tweepfake: About detecting deepfake tweets. *Plos one*, 16(5):e0251415, 2021.

[14] S. Gehrmann, H. Strobelt, and A. M. Rush. Gltr: Statistical detection and visualization of generated text. *arXiv preprint arXiv:1906.04043*, 2019.

[15] F. Gilardi, M. Alizadeh, and M. Kubli. Chatgpt outperforms crowd-workers for text-annotation tasks. *arXiv preprint arXiv:2303.15056*, 2023.

[16] B. Guo, X. Zhang, Z. Wang, M. Jiang, J. Nie, Y. Ding, J. Yue, and Y. Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*, 2023.

[17] X. He, Z. Lin, Y. Gong, A. Jin, H. Zhang, C. Lin, J. Jiao, S. M. Yiu, N. Duan, W. Chen, et al. Annollm: Making large language models to be better crowdsourced annotators. *arXiv preprint arXiv:2303.16854*, 2023.

[18] M. Heikkilä. Why detecting ai-generated text is so difficult (and what to do about it), 2023. Accessed on June 29, 2023.

[19] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[20] D. Ippolito, D. Duckworth, C. Callison-Burch, and D. Eck. Automatic detection of generated text is easiest when humans are fooled. *arXiv preprint arXiv:1911.00650*, 2019.

[21] M. Jakesch, J. T. Hancock, and M. Naaman. Human heuristics for ai-generated language are flawed. *Proceedings of the National Academy of Sciences*, 120(11):e2208839120, 2023.

[22] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023.

[23] J. Kocoń, I. Cichecki, O. Kaszyca, M. Kochanek, D. Szydło, J. Baran, J. Bielaniewicz, M. Gruza, A. Janz, K. Kanclerz, et al. Chatgpt: Jack of all trades, master of none. *arXiv preprint arXiv:2302.10724*, 2023.

[24] T. Kumarage, J. Garland, A. Bhattacharjee, K. Trapeznikov, S. Ruston, and H. Liu. Stylometric detection of ai-generated text in twitter timelines. *arXiv preprint arXiv:2303.03697*, 2023.

[25] Y. Liao, X. Jiang, and Q. Liu. Probabilistically masked language model capable of autoregressive generation in arbitrary word order. *arXiv preprint arXiv:2004.11579*, 2020.

[26] E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, and C. Finn. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *arXiv preprint arXiv:2301.11305*, 2023.

[27] R. OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023.

[28] C. Qin, A. Zhang, Z. Zhang, J. Chen, M. Yasunaga, and D. Yang. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*, 2023.

[29] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[30] T. Ryan-Mosley. Junk websites filled with ai-generated text are pulling in money from programmatic ads, 2023. Accessed on July 1, 2023.

[31] V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang, and S. Feizi. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*, 2023.

[32] M. Sadeghi and L. Arvanitis. Rise of the newsbots: Ai-generated news websites proliferating online, 2023. Accessed on July 1, 2023.

[33] I. Solaiman, M. Brundage, J. Clark, A. Askell, A. Herbert-Voss, J. Wu, A. Radford, G. Krueger, J. W. Kim, S. Kreps, et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*, 2019.

[34] R. Tan, B. A. Plummer, and K. Saenko. Detecting cross-modal inconsistency to defend against neural fake news. *arXiv preprint arXiv:2009.07698*, 2020.

[35] P. Törnberg. Chatgpt-4 outperforms experts and crowd workers in annotating political twitter messages with zero-shot learning. *arXiv preprint arXiv:2304.06588*, 2023.

[36] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[37] A. Uchendu, Z. Ma, T. Le, R. Zhang, and D. Lee. Turingbench: A benchmark environment for turing test in the age of neural text generation. *arXiv preprint arXiv:2109.13296*, 2021.

[38] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi. Defending against neural fake news. *Advances in neural information processing systems*, 32, 2019.

[39] X. Zhao, P. Ananth, L. Li, and Y.-X. Wang. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*, 2023.

[40] X. Zhao, Y.-X. Wang, and L. Li. Protecting language generation models via invisible watermarking. *arXiv preprint arXiv:2302.03162*, 2023.

Storage Systems: Organization, Performance, Coding, Reliability, and Their Data Processing, 1st Edition, October 13, 2021, Author: Alexander Thomasian

A. Thomasian
Thomasian and Associates
Pleasantville, NY
alexthomasian@gmail.com

Why This Book?

Textbooks on computer organization and architecture, operating systems, and databases do not sufficiently cover storage systems. This is especially so in view of rapid advances in storage technology in the form of flash and *Storage Class Memories - SCMs* and storage organizations such as data-centric and the DisAggregated Shared Everything - DASE.

Garth Gibson's PhD thesis provides the RAID classification, but mainly deals with reliability modeling [1]. RAID is also a proposal to replace expensive, large form factor, high capacity disks associated with mainframe computers known as Direct Access Storage Devices - DASD with an array of low cost, small form factor and capacity disks used in personal computers. At this time all high capacity *Hard Disk Drives - HDDs* have a small form factor with *Fixed Block Architecture - FBA* with 4 KB sectors. The last DASD was the IBM 3390 was last produced in May 1993.

Only 6 out of 105 pages on storage systems are dedicated to RAID in the latest 6th edition of Hennessey and Patterson Computer Architecture book [2]. A collection of 45 "influential" papers on storage systems is presented in [3], but given that most readers have access to digital libraries about 1700 references are provided in the reviewed book [4].

Storage systems are reviewed from different viewpoints as reflected by the book's title. Given the rapid evolution in the field readers are encouraged to follow storage conferences such as USENIX's File and Storage Technologies - FAST and Mass Storage and Technology - MSST:
`https://www.usenix.org/conference/fast23/storage/conference.us` and trade publications such as:
`https://blocksandfiles.com/2023/07/05/legacy-external-storage/`
`https://www.theregister.com/2020/10/09/key_role_of_storage/`

Desirable attributes for storage systems are high capacity, low latency, high data access rates (randomly placed data blocks), high data transfer bandwidth (sequential data), no hiccups (write amplification, writing to flash delayed by garbage collection, writing onto HDDs with *Shingled Magnetic Recording - SMR*), *Flash Translation Layer - FTL* to balance wear and picking blocks, nonvolatility (achieved by Uninterruptible Power Supply - UPS), shock resistance, absence of mechanical parts, low cost per GB, and low power consumption. durability (improved by RAID level), lifespan (HDDs and NAND flash SSD, 3-5 years, M-disc 1000 years).

Low latency and high access rates are important in *On-Line Transaction Processing - OLTP* and e-commerce applications, which require high access rates to randomly placed disk blocks. Data mining requires high transfer rates to access large volumes of data, as in the case of Birch data clustering method [4]. Very high archival storage capacity is achievable by robotic libraries of magnetic tapes.

While hyperscalers (large disk farms) are based on HDDs, a second paradigm shift is due to flash *Solid-State Drives - SSDs* replacing HDDs due to lower access time and high bandwidth. Lower power consumption and higher reliability resulting in a lower TCO.

Flash SSDs have lower access times and bandwidths: DRAM: 75 ns, 2.4 GB/s; SCM 179 ns, 2.4 GB/s; SSD 85 μs (microseconds), 1.6 GB/s; HDD mean seek time plus T_{rot} = [3600/RPM]/2 divided by two, 100-200 MB/s. The transfer time of small blocks tends to be negligibly short. SCMs serve as cache frontends to SSDs reducing flash wearout.

One intent of the text is to familiarize the reader with storage technologies used by computers. As the volume of data is increasing exponentially storage efficiency can be increased via lossless data compression, which depends on data category: text (Huffman, Lempel-Ziv), audio, images (JPEG), video (MPEG). Data deduplication is used to reduce the volume of data for archiving. Data encryption is used for higher data security and privacy.

The RAID paradigm utilizes replication or erasure coding to deal with failures. Data is replicated at multiple distributed sites to deal with disasters: earthquakes, floods, fires, power outages, but additional protection is required against ransomware, computer viruses.

Most data processing used to be carried out using disk-resident files and later databases. Given complex SQL queries higher processing efficiency was achievable by selecting appropriate indexing and applying sophisticated query optimization methods. *Log Structured Merge Trees - LSMTs* are more appropriate than B+ trees in SSDs. New algorithms have been developed for NoSQL, multimodel, document, graph, key-value stores. Current research is now focused on data processing on flash and other storage technologies.

Maximum Disk Separable - MDS RAID$(4+k), k \geq 1$ disk arrays use the minimum redundancy level, k redundant disks to tolerate k disk failures. RAID6 utilizes RS codes, but IBM's EVENODD and NetApp;s Row-Diagonal Parity - RDP are two parity–based implementations incurring lower computational cost than RS. Both have been extended to 3DFTs. Similarly RAIDIX RAID7.3 uses $k = 3$ parity disks

to protect 45 data HDDs.

When $j \leq k$ disks fail in RAID(4+K) with $N+k$ disks fail their contents can be reconstructed on demand by accessing the corresponding blocks on N disks. Noting that with j failed disks the read load is increased (j+1)-fold so provided sufficient spare space is available rebuild processing should be started right away. With $j = k$ disk failures the system is in a critical state, since more failure will lead to data loss.

A common reason for data loss during rebuilds are *Latent Sector Errors - LSEs*, which can be dealt with disk scrubbing and IntraDisk Redundancy - IDR. In [8] we review techniques based on ML which predict disk failures.

In the case of RAID6 preferably two failed disks are rebuilt simultaneously, rather than one disk at a time. Rebuild processing in multi-TB HDDs can takes many hours and there is the possibility of data loss due to additional disk failures, especially if they are correlated.

Pyramid codes use Local Redundancy Codes - LRC for low cost recovery and they were later utilized by Windows Azure distributed storage system to reduce communication cost.

Data Base Machines - DBMs use parallel processing to achieve high-performance DB processing. Tandem/HP NonStop SQL and Teradata DBC/1012 both initially used mirrored disks for higher reliability.

Two chapters are dedicated to author's research dealing with Heterogeneous Disk Arrays - HDAs, Hierarchical RAID, and related studies.

Intended Audience and Background Requirements

The book is intended for upper division undergraduate and master level computer science and engineering, electrical engineering, and data and information science and technology students and professionals. The book is interspersed with discussion of performance and reliability analysis.

At this time computing permeates all areas of engineering, science, technology, and business hence knowledge of computer hardware and software and associated companies is necessary for all professionals even students choosing a laptop for high school.

Background in basic computer science, mathematics (discrete, probability, and Markov chains), Brief tutorial material is provided on reliability modeling and queueing theory.

In addition to references in the bibliography a list of relevant books and various sources is provided in the Appendix.

Book Chapters

The book TOC has a full list of book topics.

Chapter 1: Introduction. We start with history of companies and their computers in commercial, scientific, engineering, and military applications after WW II. IBM introduced the S/360 computer family in 1964 which led in 1967 for to it to become the most highly valued company S/360 computers now called IBM Zsystems were extended with caches, virtual memory, 64 bit addressing, data compression, encryption. Companies in the BUNCH (Burroughs, Univac, NCR, CDC, Honeywell) plus DEC and HP had their own proprietary computers and software, which has since disappeared while Wintel - Microsoft Windows and Intel are dominant. HDDs are produced by Seagate, Western Digital etc. and disk arrays: Dell/EMC, HPE, NetApp, etc.

Chapter 2: Storage Technologies. Paper allowed writing and later printing. Punched cards were used to store census data, company records and simple data processing. Magnetic tapes and disks (also optical) held data, images, audio and video. Tapes and disks were used for batch and disks with random access capabilities for online data processing. Tapes only accessible sequentially are used for batch processing, while disks in allow random access via hashing or indexing useful for OLTP Extensible and linear hashing allows dynamic additions and deletion of records and so do B+trees/VSAM. For high performance flash memories and other SCMs - Storage Class Memories such as PCM - Phase Change Memory are expected to replace magnetic disks. Data compression and deduplication can be used in preserving storage space and network bandwidth and encryption for data security and privacy.

Chapter 3: Disk drive organization, data Placement, and scheduling. Commodity disks with fixed size sectors replaced CKD DASD associated with IBM mainframes. For mainframes running legacy application under z/OS CKD DASD are simulated by the disk array controllers running z/OS on IBM mainframes. Zoned bit recording - ZBR, Shingled Magnetic Recording - SMR and Energy Assisted Magnetic Recording - EAMR provide higher storage capacity. Disk scheduling policies and data placement to minimize seek and access time. Shortest Access Time First - SATF is extended with lookahead and two priority classes. Queueing analysis of disk performance with FCFS (with Zoned Bit Recording - ZBR), SCAN, and SATF policies is discussed. A review RAID performance analyses using queueing theory and simulation is presented.

Chapter 4: Mirrored and hybrid arrays. Mirrored disks, classified as RAID1, protect against disk failures by storing data twice. In a hybrid disk array with half of the disks hold data and the other half *eXclusive ORed - XORed* data: $\text{Disk}_{2n} = \text{Disk}_{2n-1} \oplus \text{Disk}_{(2n+1)\text{mod}(12)}, 1 \leq n \leq 6$. This allows all two disk failures and half of consecutive three disk failures to be tolerated, while there s data loss with basic Mirroring when a disk pair fails. RAID1 doubles disk access time bandwidth, but when a disk fails the load on the surviving disk is doubled. RAID1 configurations which distribute the load of a failed disk over multiple disks achieve a more balanced load. Shortcut reliability analysis shows that BM is most reliable RAID1, but less reliable than hybrid disk arrays Analytic and simulation methods to estimate RAID reliability are presented.

Chapter 5: Variation of RAID with emphasis on RAID(4+k), k=1,2,3. Operation in normal, degraded, and rebuild modes is discussed and their performance analyzed. Clustered RAID5 has a parity group size $G < N$, where N is the number of disks. *Balanced incomplete Block Designs - BIBD*, Thorp shuffle, *Nearly Random Permutations - NRP*, and row shifting are four methods to place parities balance updating of disk loads. The Vacationing Server Model - VSM for RAID5 is analyzed/ It processes rebuilds requests only when the queue of external requests is empty. VSM outperforms the Permanent Customer Model - PCM in rebuild time and external request response time degradation. IntraDisk redundancy - IDR and disk scrubbing methods to deal with Latent Sector Errors - LSEs, which may lead to unsuccessful rebuilds are presented. IDR is a low cost method in terms of space ad performance which obviates the need for higher redundancy levels than RAID5.

Chapter 6: Coding Methods. EVENODD, RDP, and X-code are Maximum Distance Separable - MDS codes in that they incur the minimum level of redundancy to tolerate two disk failures with parity coding. RDP is shown to outperform EVENODD in the number of XORs required as shown by M. Blaum.

Chapter 7: Power Reduction in Storage Systems and Servers. Power consumption is a major problem for server and storage clouds. Server energy can be saved by lowering voltage and in the case of disks by lowering RPMs and spindown. In the case of laptops there is a switch to SSDs which use less power. Massive Arrays of Independent Disks - MAID paradigm leaves a few disk spinning for caching recently accessed data and for new updates. Hybrid disks use a flash memory or NVRAM cache to hold updates while the disk is spundown. Power consumption can be reduced by temporal and spatial clustering of disk accesses. Submerging servers into liquid baths is a new development.

Chapter 8: Database parallelism, big data, analytics, and deep learning. Early Data Base Machines - DBMs mainframes running database management systems, such as IMS. Data mining such as Association Rule Mining on mainframes was considered too costly, since results such as diapers and beer bought together was of questionable value. The advent of low-cost powerful microprocessors allowed the building of highly parallel DBMs for dealing with big data.

Active disks processed high priority disk accesses for OLTP, while processing data mining requests at no cost by freeblock scheduling. Processor per track disks such as the Relational Associative Processor - RAP, are no longer feasible because of high track densities, but this paradigm has been applied to FAWN - FAST Array of Wimpy (flash) Nodes and RamCloud which attains durability by keeping backup copies on secondary storage. It was predicted by Gibson in [1] that DRAM will replace HDDs after 2027.

Extracting information from data for decision making utilizes CPUs, GPUs, FPGAs, Application Specific Integrated Circuits - ASICs, and Tensor Google's Processing Units - TPUs, which is Google's ASIC for *Machine Learning - ML.* with Deep Neural Networks. DPUs - Data Processing Units combine a CPU, GPU, and a network interface.

Chapter 9: Structured, semi-structured, and unstructured data. These include network, hierarchical, and relational databases, big data, Hadoop technology ecosphere, NoSQL databases, key-value stores, document stores, times series, graph, and object-oriented databases, web search engines, Resource Description Framework - RDF, wide column stores, native XML and JSON databases, realtime database, event stores, content store, multimodel databases, main memory databases. Novel applications such as streaming analytics and business intelligence platforms.

Chapter 10: Heterogeneous Disk Arrays - HDAs. allow multiple Virtual Arrays - VAs to share disk space using controllers for multiple RAID levels. The RAID level and organization of VAs is determined by reliability and workload requirements, e.g., RAID1 suited for OLTP with high access rates to small data blocks, while RAID5 suited for parallel accesses to large files. Allowing variable RAID levels conserves disk bandwidth, because not all VAs need to be allocated with the highest redundancy level required by a subset of datasets.

Chapter 11: Hierarchical RAID. HRAID is motivated by the storage bricks paradigm. There are N Storage Node - SN whose controllers implement the intraSN and interSN code with k and ℓ check codes. Thus HRAID uses a multilevel RAID paradigm with interSN erasure coding.

Disk failures are handled by intraSN redundancy firstly and more costly interSN redundancy secondly. An baseline array with no interdisk redundancy is considered for comparison purposes. Operations with and without interSN rebuild processing is considered in simulations, which are also used to determine the effect of varying HRAID parameters on the MTTDL - Mean Time to Data Loss. We disagree on how redundancy is apportioned in IBM's Intelligent Bricks project in [6].

Chapter 12: Some Further Topics. This is to encourage readers to pursue them on their own.

An extension of the seminal RAID tutorial [7] appears in [8] extending some discussions in [5].

References

[1] G. A. Gibson. Redundant Disk Arrays: Reliable Parallel Secondary Storage. The MIT Press 1992.

[2] J. L. Hennessey and D, Patterson. Computer Architecture: A Quantitative Approach, 6th edition. Elsevier 2019.

[3] H. Jin, B. Rajkumar and T. Cortes. High Performance Mass Storage and Parallel I/O: Technologies and Applications IEEE & Wiley Interscience, 2002

[4] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A new data clustering algorithm and its applications. J. Data Mining and Knowledge Discovery, 1, 2 (1997), 141182.

[5] A. Thomasian. Storage Systems: Organization, Performance, Coding, Reliability, and Their Data Processing, 1st Edition, Oct. 2021, 712 pages
Paperback ISBN: 9780323907965, eBook ISBN: 9780323908092
`https://shop.elsevier.com/books/storage-systems/`
Available chapter by chapter from ScienceDirect.
`https://www.sciencedirect.com/book/9780323907965/storage-systems`
Google books preview, 151 pages (about 20% of the book).
`https://www.google.com/books/edition/Storage_Systems/t8wnEAAAQBAJ`

[6] A. Thomasian. Optimizing Apportionment of Redundancies in Hierarchical RAID. `https://arxiv.org/pdf/2205.06330.pdf`, May 2022.

[7] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz and D. A. Patterson. RAID: High-Performance, Reliable Secondary Storage. ACM Computing Surveys 26, 2 (1994), 145-185.

[8] A. Thomasian. RAID Organizations for Improved Reliability and Performance: A Not Entirely Unbiased Tutorial. `https://arxiv.org/abs/2306.08763`, June 2023.

Alexander Thomasian, Life Fellow, IEEE

With a BSEE Degree from Univ. of Tehran I joined IBM WTC as a Systems Engineer. At Tehran's electric utility developed the billing application for 1/2 million customers. With a CS PhD from UCLA I taught at Case Western Reserve and then Univ. Southern Calif. At Almaden Research Center I analyzed the performance of IBM's RAID5. Back in Yorktown I contributed to a NASA sponsored project for indexing satellite images. At NJIT my research was funded by NSF, Hitachi, and AT&T's Virtual Univ. Research Inst. I was at `siat.cas.cn` (2010-11) and `aua.am` as a Fulbright Fellow (2015). I have four patents, over 150 papers, a book republished by Springer on concurrency control, and supervised ten PhD students.

Report on the 3rd International Workshop on Learning to Quantify (LQ 2023)

Mirko Bunse[1], Pablo González[2], Alejandro Moreo[3], and Fabrizio Sebastiani[3]
[1] Lamarr Institute for Machine Learning and Artificial Intelligence,
TU Dortmund University, 44227 Dortmund, DE
[2] Artificial Intelligence Center, University of Oviedo, 33204 Gijón, ES
[3] Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, 56124 Pisa, IT
mirko.bunse@cs.tu-dortmund.de, gonzalezgpablo@uniovi.es,
alejandro.moreo@isti.cnr.it, fabrizio.sebastiani@isti.cnr.it

ABSTRACT

The 3rd International Workshop on Learning to Quantify (LQ 2023)[1] took place on September 18, 2023 in Torino, IT, where it was organised as a satellite event of the 34th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2023). Like the main program of the conference, the workshop employed a hybrid format, with all presentations given in presence and with attendees participating in presence or online. This report presents a summary of the workshop, briefly summarising the individual works presented, and touching on the main issues that emerged during the final, open discussion.

1. INTRO: LEARNING TO QUANTIFY

Many fields such as the social sciences, political science, market research, or epidemiology (to name a few), are inherently interested in *aggregate* data, i.e., in how populations of individuals are distributed according to one or more indicators of interest. Researchers who operate in these specialty areas are instead little interested in the individual data items *per se*, since in these fields the individual data items are relevant only inasmuch as they are members of the population of interest; in other words, disciplines such as the above are interested not in finding the needle, but in characterising the haystack.

Sometimes, researchers active in these fields use supervised learning to obtain the data they need. For instance, epidemiologists interested in the distribution of the causes of death across different geographical regions may sometimes need to *infer* the cause of death of each person by classifying, via a machine-learned text classifier, a verbal description of the symptoms that affected a deceased person. However, epidemiologists are not specifically interested in the class (representing a given cause of death) to which an individual belongs; rather, their final goal is estimating the *prevalence* (i.e., *relative frequency*, or *prior probability*) of each class in the unlabelled data.

At a first glance, estimating these prevalence values via supervised learning looks like a direct application of classification, since one could simply (i) train a classifier on labelled data, (ii) use this classifier to issue label predictions for each unlabelled datapoint (i.e., for each individual) in the population of interest, (iii) count how many datapoints have been attributed to each of the classes of interest, and (iv) normalize the counts by the total number of labelled datapoints, thus obtaining the estimated relative frequencies of the classes. However, there is by now abundant evidence that such an approach, known in the literature as the "Classify and Count" (CC) method, yields poor class prevalence estimates when the distribution of the unlabelled datapoints across the classes differs substantially from the analogous distribution observed during training [12]. This latter condition is typically known as *prior probability shift* (or *label shift*); in the aforementioned disciplines this condition is ubiquitous since, quite obviously, there is interest in inferring a distribution *only* if we consider the possibility of this unknown distribution to be different from the known distribution of the training data. The main reason for which CC tends to fail in the presence of prior probability shift is a violation of the IID assumption on which many supervised learning methods are based.

Due to the suboptimality of CC, learning to quantify has slowly evolved as a task in its own right, different from classification in terms of goals, methods, techniques, and evaluation measures [9, 13]. The research community has investigated methods to correct the biased prevalence estimates of general-purpose classifiers, supervised learning methods specifically tailored to LQ, and evaluation measures for LQ. Applications of LQ have also been investigated, such as sentiment quantification, quantification in networked environments, or quantification for data streams. For the near future, it is easy to foresee that the interest in learning to quantify will increase, due (a) to the increased awareness that "classify and count" is a suboptimal solution when it comes to prevalence estimation, and (b) to the fact that, with larger and larger quantities of data becoming available and requiring interpretation, in more and more scenarios we will only be able to afford an analysis of these data at the aggregate level rather than at the individual level. At ECML/PKDD 2023, the increase in awareness about LQ manifested in an LQ paper [8] being given the best student paper award.

[1] https://lq-2023.github.io/

2. THE WORKSHOP

LQ 2023 was a combined tutorial + workshop event, i.e., it consisted of a half-day (morning) tutorial on quantification (which was taught by the third and fourth authors of the present paper) plus a half-day (afternoon) workshop. The tutorial introduced the rationale for quantification (also discussing the relationship between quantification and various types of dataset shift), discussed a number of domains to which quantification has been applied to, presented the main evaluation measures and the main experimental protocols that have been used in the quantification literature, and gave an overview of the main classes of methods that have been used for performing quantification. The tutorial also included a "hands-on" session, which made use of the QuaPy open-source Python library for quantification developed by the instructors [18]. The workshop consisted instead of the oral presentations of seven papers submitted in response to the call for papers, plus a final brainstorming session, in which the perceived "burning issues" of the field were brought up and discussed. The combine event was attended by about 30 people, of which about 20 in-presence and about 20 online.

2.1 The papers

The call for papers had asked for either completely original papers *or* papers that had recently (i.e., in 2023) been submitted / accepted / published in other workshops / conferences / journals; as a result, 3 papers of the first kind [4, 15, 21] and 4 papers of the latter kind [6, 7, 11, 19] were accepted for presentation at the workshop. The seven contributed talks were selected by a program committee consisting of 12 renowned LQ experts; each paper was reviewed by at least 3 members of the committee.

Dirk Tasche presented his work "Invariance assumptions for class distribution estimation" [21], which discusses various assumptions of invariance between the distributions of training and test data (covariate shift, factorizable joint shift, and sparse joint shift) and the implications that each of these assumptions has on quantification learning. While factorizable joint shift is shown to yield no consistent estimator of class prevalence values, sparse joint shift is shown to generalize prior probability shift, the type of shift which is most frequently assumed in quantification learning.

Gustavo Batista presented "MC-SQ and MC-MQ: Ensembles for multi-class quantification", a joint work with Zahra Donyavi and Adriane Serapião that was the subject of a recent conference publication [7]. The presentation started by noting that tackling multi-class quantification via the one-versus-all strategy (consisting of training an ensemble of independent binary quantifiers, one per class) generally yields poor performance. The authors thus propose *multiple-classifiers single-quantifier* (MC-SQ) and *multiple-classifiers multiple-quantifiers* (MC-MQ), two new ensemble-based models for quantification that couple different classifiers with different aggregative quantifiers. The experiments presented showed that MC-SQ and MC-MQ are new important contenders in the multiclass quantification arena.

Alessandro Fabris presented "Measuring Fairness under Unawareness of Sensitive Attributes: A Quantification-Based Approach", a joint work with Andrea Esuli, Alejandro Moreo, and Fabrizio Sebastiani that was the subject of a recent journal publication [11]. In this paper, the authors use a method based on quantification in order to measure how fair (or how biased) a classifier is with respect to a given sensitive attribute (e.g., race, sex) which is not among the attributes (i.e., covariates) used by the classifier ("unawareness"). The authors conclude that the method has two important advantages, i.e., (a) it drastically outperforms methods based on standard classification, and (b) it manages, differently from methods based on classification, to obtain the desired aggregate-level predictions while not allowing undesired predictions at the individual level.

Mirko Bunse presented his work "Qunfold: Composable Quantification and Unfolding Methods in Python" [4], which documents a novel software package for quantification methods. Building on the findings of two works that the author presented at the 2nd edition of the LQ workshop, this easy-to-use package allows its users to compose novel quantification methods from existing loss functions and data representations [3], and it implements a powerful optimization technique [2] that the author shows to achieve lower prediction errors than other state-of-the-art implementations.

Pablo González presented "An Equivalence Analysis of Binary Quantification Methods", a joint work with Alberto Castaño, Jaime Alonso, and Juan J. del Coz. In this paper the authors analyse several algorithms falling under the distribution-matching framework [17], finding theoretical connections and equivalences between them. They also propose a new method called "QUANTy", based on average probabilities computed in different quantiles. The authors conclude by emphasizing the importance of using richer representations for characterising the distributions (in contrast to simplistic representations that may lose important information about them). The paper does not appear in the proceedings of LQ 2023 but is available as [6].

Kevin Kloos presented "Continuous Sweep: An Improved Binary Quantifier", a joint work with Julian D. Karch, Quinten A. Meertens, and Mark de Rooij. In this paper the authors propose a binary quantification method called "Continuous Sweep", based on the well-known "Median Sweep" quantification algorithm [12]. Continuous Sweep is a quantifier that uses parametric class distributions instead of empirical distributions. The paper presents theoretical derivations for the bias and variance of the method; simulation studies show that it outperforms its predecessor on datasets characterised by prior probability shift. The paper does not appear in the proceedings of LQ 2023 but is available as [15].

Alejandro Moreo presented the paper "Multi-Label Quantification", a joint work with Manuel Francisco and Fabrizio Sebastiani that was the subject of a recent journal publication [19]. In this paper the authors systematically investigate (by focusing on "aggregative" quantification) the case in which the data items can each have zero, one, or several labels at once (a task that had never been investigated before). The conclusion of their study is that methods that try to exploit the stochastic correlations among the classes *both* in the underlying classifier *and* in the aggregation policy, outperform all aggregative methods that either exploit these correlations in one of the two phases only or do not exploit them at all. The authors indeed propose two new methods that do exploit these correlations; an interesting aspect of these two methods is that they can also be used in conjunction with non-aggregative quantification methods.

2.2 The discussion

The workshop ended with a brainstorming session, which had the double purpose of acting as an overflow space for all those questions for which there had been no time during the Q&A sessions of the papers, and of allowing the LQ community to discuss, in an unconstrained, informal setting, what the participants perceived to be the "burning", still unresolved issues in this field.

2.2.1 Datasets for evaluating quantification

One of the major issues discussed was the lack of datasets for ideally supporting the evaluation of quantification. Most current datasets (including the one used in the recent LeQua 2022 data challenge, which was specifically devoted to quantification [10]) originated as classification datasets (i.e., are composed of a training set and a test set of individual labelled items), and were turned into quantification datasets by extracting from the test set, according to a certain extraction protocol, a number of test samples meant to cover the entire spectrum of prevalence distributions.

One consequence of this fact is that quantification research has been carried out on too many different datasets (as opposed to a few "standard" datasets) since, in the absence of true quantification datasets, it was easy for each author to pick her/his favourite classification dataset and turn it into a quantification dataset via a suitable extraction protocol; this has led to the fact that results reported in different research papers on quantification are often not comparable with each other.

Another consequence is that, in the above type of experimentation, the testing samples, which are generated by the extraction protocol, may arguably be unrealistic, at least in some cases. For instance, when applying quantification to *land cover mapping* (LCM),[2] the above extraction protocol would involve the generation of a sample by assembling different pixels "extracted" from different images, which is clearly unrealistic. In LCM, a realistic dataset would be composed of some training images and some test images, where each image is a sample of (labelled or unlabelled) items (the pixels); this scenario is unlike those seen in most experimental evaluations of quantification systems, in which (a) the training set is a set of individual labelled items (and not a set of *samples of* individual labelled items, as in LCM), and (b) the test set is a set of *artificially generated* samples (i.e., samples extracted via the protocol) of unlabelled individual items (and not a set of *naturally occurring* such samples, as in LCM).

LCM seems to evoke the so-called *natural prevalence protocol* (NPP – [9, §3.4.1]) for experimentation in quantification, where only naturally occurring samples are tested upon; other applications of quantification, such as monitoring insect populations [7], estimating the prevalence of different species of plankton in sea water samples [14], quantifying the number of damaged cells in biological samples [20], or seabed cover mapping [1], are also characterised by this property. Unfortunately, the use of the NPP tends to be feasible only when true quantification datasets (i.e., datasets in which both training set and test set naturally come as sets of samples) are available, and public datasets of this sort tend to be hard to obtain (note that there are several LCM datasets that are publicly available, but they contain *estimates* of the pixel labels, and not the ground truths that would be necessary for evaluation).

The discussion on this topic was concluded by a collective commitment to look out for candidate datasets that consist of naturally occurring samples. Future editions of the LQ workshop should plan to support this commitment by soliciting data-centric submissions in the call for papers.

2.2.2 A LeQua competition in 2024?

A related topic of discussion was a possible follow-up to LeQua 2022, the first data challenge specifically devoted to quantification [10]. Many attendees stated that LeQua 2022 turned out to be very useful because it provided a controlled environment in which different quantification methods could be experimentally compared and because it generated a reference collection on which systems could be (and have been) tested even after the end of the official challenge.

The attendees agreed that a 2nd edition of the challenge would be invaluable in deepening our collective understanding of quantification. The discussion ended with a collective commitment to think of possible formats for this 2nd edition, of possible subtasks of quantification to focus on, of possible conferences to co-locate it with, and of ways to increase participation in the challenge.

3. CONCLUSION

Overall, LQ 2023 was a success, both in terms of participation and, above all, in terms of the livelihood of interaction among the participants that emerged in the Q&A sessions after the individual papers and in the final brainstorming session. The LQ workshop series has proven itself to be an important reference point for researchers who are active in quantification, especially due to the fact that these researchers are scattered across different research communities (statistics, information retrieval, data mining, machine learning, and others) and are thus unlikely to meet in other contexts.

The proceedings of LQ 2023 appear in a self-published form as [5], and are freely available from the LQ 2023 website (`https://lq-2023.github.io/`).

Acknowledgements

The work by the first author has been funded by the Federal Ministry of Education and Research of Germany and the state of North Rhine-Westphalia as part of the Lamarr Institute for Machine Learning and Artificial Intelligence. The work by the second author has been funded by MINECO (Ministerio de Economía y Competitividad) and FEDER (Fondo Europeo de Desarrollo Regional), grant PID2019-110742RB-I00 (MINECO/FEDER). The work by the third and fourth authors has been supported by the AI4Media and SoBigData++ projects, funded by the European Commission (Grant 951911 and 871042, respectively) under the H2020 Programme ICT-48-2020, and by the SoBigData.it, FAIR, and QuaDaSh projects funded by the Italian Ministry of University and Research under the NextGenerationEU program. The authors' opinions do not necessarily reflect those of the funding agencies.

[2]Land cover mapping is an application in which, given an aerial photograph of a portion of the Earth, one has to estimate the fractions of pixels that indicate Trees, or Shrubland, or Grassland, or other types of land cover [16].

4. REFERENCES

[1] O. Beijbom, J. Hoffman, E. Yao, T. Darrell, A. Rodriguez-Ramirez, M. Gonzalez-Rivero, and O. Hoegh-Guldberg. Quantification in-the-wild: Datasets and baselines. CoRR abs/1510.04811 (2015). Presented at the NIPS 2015 Workshop on Transfer and Multi-Task Learning, Montreal, CA, 2015.

[2] M. Bunse. On multi-class extensions of adjusted classify and count. In *Proceedings of the 2nd International Workshop on Learning to Quantify (LQ 2022)*, pages 43–50, Grenoble, IT, 2022.

[3] M. Bunse. Unification of algorithms for quantification and unfolding. In *Proceedings of the Workshop on Machine Learning for Astroparticle Physics and Astronomy*, pages 459–468, Hamburg, DE, 2022.

[4] M. Bunse. Qunfold: Composable quantification and unfolding methods in Python. In *Proceedings of the 3rd International Workshop on Learning to Quantify (LQ 2023)*, pages 1–7, Torino, IT, 2023.

[5] M. Bunse, P. González, A. Moreo, and F. Sebastiani, editors. *Proceedings of the 3rd International Workshop on Learning to Quantify (LQ 2023)*. Torino, IT, 2023.

[6] A. Castaño, J. Alonso, P. González, and J. J. del Coz. An equivalence analysis of binary quantification methods. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023)*, pages 6944–6952, Washington, US, 2023.

[7] Z. Donyavi, A. Serapio, and G. Batista. MC-SQ: A highly accurate ensemble for multi-class quantification. In *Proceedings of the 23rd SIAM International Conference on Data Mining (SDM 2023)*, pages 622–630, Minneapolis, US, 2023.

[8] B. Dussap, G. Blanchard, and B.-E. Chérief-Abdellatif. Label shift quantification with robustness guarantees via distribution feature matching. arXiv:2306.04376 [stat.ML], 2023.

[9] A. Esuli, A. Fabris, A. Moreo, and F. Sebastiani. *Learning to quantify*. Springer Nature, Cham, CH, 2023.

[10] A. Esuli, A. Moreo, F. Sebastiani, and G. Sperduti. A detailed overview of LeQua 2022: Learning to quantify. In *Working Notes of the 13th Conference and Labs of the Evaluation Forum (CLEF 2022)*, Bologna, IT, 2022.

[11] A. Fabris, A. Esuli, A. Moreo, and F. Sebastiani. Measuring fairness under unawareness of sensitive attributes: A quantification-based approach. *Journal of Artificial Intelligence Research*, 76:1117–1180, 2023.

[12] G. Forman. Quantifying trends accurately despite classifier error and class imbalance. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, pages 157–166, Philadelphia, US, 2006.

[13] P. González, A. Castaño, N. V. Chawla, and J. J. del Coz. A review on quantification learning. *ACM Computing Surveys*, 50(5):74:1–74:40, 2017.

[14] P. González, A. Castaño, E. E. Peacock, J. Díez, J. J. Del Coz, and H. M. Sosik. Automatic plankton quantification using deep features. *Journal of Plankton Research*, 41(4):449–463, 2019.

[15] K. Kloos, J. D. Karch, Q. A. Meertens, and M. de Rooij. Continuous Sweep: An improved, binary quantifier. arXiv:2308.08387 [stat.ML], 2023.

[16] P. Latinne, M. Saerens, and C. Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities may significantly improve classification accuracy: Evidence from a multi-class problem in remote sensing. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 298–305, Williamstown, US, 2001.

[17] A. Maletzke, D. Moreira dos Reis, E. Cherman, and G. Batista. DyS: A framework for mixture models in quantification. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, pages 4552–4560, Honolulu, US, 2019.

[18] A. Moreo, A. Esuli, and F. Sebastiani. QuaPy: A Python-based framework for quantification. In *Proceedings of the 30th ACM International Conference on Knowledge Management (CIKM 2021)*, pages 4534–4543, Gold Coast, AU, 2021.

[19] A. Moreo, M. Francisco, and F. Sebastiani. Multi-label quantification. *ACM Transactions on Knowledge Discovery and Data*, 18(1):Article 4, 2023.

[20] L. Sánchez, V. González, E. Alegre, and R. Alaíz. Classification and quantification based on image analysis for sperm samples with uncertain damaged/intact cell proportions. In *Proceedings of the 5th International Conference on Image Analysis and Recognition (ICIAR 2008)*, pages 827–836, Póvoa de Varzim, PT, 2008.

[21] D. Tasche. Invariance assumptions for class distribution estimation. In *Proceedings of the 3rd International Workshop on Learning to Quantify (LQ 2023)*, pages 56–71, Torino, IT, 2023.

Anomaly Detection using Generative Adversarial Networks

Reviewing methodological progress and challenges

Fiete Lüer
eMundo GmbH, Gofore Oyj
Hofmannstr. 25-27
Munich, Germany
fiete.lueer@e-mundo.de

Christian Böhm
University of Vienna
Währinger Straße 29
Vienna, Austria
christian.boehm@univie.ac.at

ABSTRACT

The applications of Generative Adversarial Networks (GANs) are just as diverse as their architectures, problem settings as well as challenges. A key area of research on GANs is anomaly detection where they are most often utilized when only the data of one class is readily available.
In this work, we organize, summarize and compare key concepts and challenges of anomaly detection based on GANs. Common problems which have to be investigated to progress the applicability of GANs are identified and discussed. This includes stability and time requirements during training as well as inference, the restriction of the latent space to produce solely data from the normal class distribution, contaminated training data as well as the composition of the resulting anomaly detection score. We discuss the problems using existing work as well as possible (partial) solutions, including related work from similar areas of research such as related generative models or novelty detection. Our findings are also relevant for a variety of closely related generative modeling approaches, such as autoencoders, and are of interest for areas of research tangent to anomaly detection such as image inpainting or image translation.

Keywords

Adversarial Generative Models, Anomaly Detection, Generative Adversarial Network, Novelty Detection, Outlier Detection

1 Introduction

Anomalies are commonly described as patterns in data not conforming to expected behavior [1]. Detecting anomalies is a frequent problem occuring on various types of data where the "expected behavior" is usually represented by a set of *normal* instances. Anomaly detection can often be framed as a binary classification problem where the task is to distinguish solely between normal and abnormal instances. Detecting anomalies is crucial for an abundance of industries: It is important to detect intrusions into networks [2], abnormal data in (spatio-temporal) climate data [3], patterns which indicate human diseases [4] or to minimize the risk of fraud [5]. In many of these applications, prompt actions are required to diminish or avoid damage, or to enable novel applications. For many such applications, recent progress has shifted the focus to deep learning algorithms with a major application being complex high dimensional data, e.g. image data, where handcrafting features is prone to errors. Neural networks can be used to detect anomalies in various ways. It is possible to compare the anomalous input data in a discriminative way using a threshold output score or forecasted values. Discriminative models such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) have found frequent use but can suffer from a variety of problems. These issues include differences between the predicted probability estimate of a class label and the ground truth correctness likelihood, which can be improved by calibrating the output [6]. Other challenges cannot easily be solved by improvements of the processing or training itself. Requiring a balanced dataset to avoid a shift to classifying only the predominant class is one of the most impactful problems. This led to a surge in generative modeling approaches such as (variational) autoencoders (VAEs) or Generative Adversarial Networks (GANs): Through one-class (OC) learning of only the normal class distribution, we can attempt to learn a generative procedure which should only produce normal class data, leveraging the reconstruction error between real and synthetic data.
For either method, an anomaly score is often calculated, e.g. based on the output score of discriminative networks or the difference between real and forecasted or synthetic values. This score can be compared to a predefined threshold which may not be exceeded by normal class input data (e.g. autoregressive neural networks [7]). Since the concept is based on learning a distribution from which specific data is generated, it can be applied to related models such as (V)AEs whose applicability we further discuss in the following sections.
Existing reviews [8, 9] on GAN-based anomaly detection focus on applications and types of data as well as architectures and metrics. Our work especially targets the investigation of more general challenges of GAN-based anomaly detection, which are mostly independent of the application, type of data or network architecture. Existing work on solving these challenges is clustered into distinctive categories and connected to related work in similar domains such as AEs.
We introduce foundations on anomaly detection as well as GANs in Section 2. The basic idea of AnoGAN [4], a central approach used to detect anomalies using GANs, is presented in Section 3. Subsequent advances and challenges are used to discuss and extend this framework in Section 4, focusing on practical as well as theoretical challenges. This especially includes i) the speed and quality of the training process, ii)

restrictions to the latent space, iii) contaminated training data, iv) novel anomaly score components and compositions to better extract relevant information as well as v) inference accuracy and speed. Lastly, this work is summarized in a general discussion in Section 5.[1]

2 Preliminary

2.1 Anomaly detection

An outlier, also called anomaly, intuitively is an observation deviating so much from other observations as to arouse suspicions that it was generated by a different mechanism [10]. The detection of anomalies is especially important in the medical domain where recent advances, mostly through novel deep learning approaches, have significantly improved the performance on various tasks and types of medical imaging. This includes MRI segmentation [11], CT scan generation [12] or Alzheimers prediction using F-FDG PET scans [13]) but also extends to time series data, sometimes even surpassing the performance of human professionals (e.g. on ECG data [14]). Due to its relevance, anomaly detection has been researched and reviewed for several decades. This includes very broad reviews [1], as well as surveys which focus on more specific data structures, such as graphs [15], where recent advances on Graph Neural Networks [16] open up interesting future applications.

Anomaly detection is closely related to novelty detection, i.e. the problem of finding novel data points. While the problem formulation and methodology *can* differ, there is a variety of tasks where it does not differ. Abati et al. [17] introduce novelty detection as the discrimination of observations that do not conform to a learned model of regularity, which can be translated to the detection of data points with novel, additional informational value. Abati et al. further argue that the reconstruction error or discriminative in-distribution tests express how well we *remember* an event and the *surprisal* is modeled by low probabilities of events under an expected model, which might be a network conditioned on normal samples, or by lowering a variational free energy. But just as in human memory, the rememberance itself might not be sufficient. Most anomaly detection tasks have a non-trivial and non-symmetric distribution of samples which can lead to only a small allowed margin of reconstruction errors in some part of a resulting learned manifold and a larger margin in another part which has to be accounted for in most applications to create a realiable convex hull. This is a major obstacle for OC anomaly detection where the manifold of normal class data, which corresponds to such "a learned model of regularity", is approximated.

The broad variety of anomaly detection methods itself ranges from fixed threshold values over more traditional machine learning approaches such as density based methods like Kernel Density Estimators [18] to more recent deep learning approaches, which are often based on reconstruction losses, e.g. autoregressive predictions of future values [7] or GANs. One of the advantages of reconstruction-based anomaly detection is that it allows the localization of the anomalies within high dimensional data such as images [19] which enables the application of further processing techniques such as segmentation or inpainting tasks. Furthermore, the quality of the reconstruction, measured by the degree it differs from the test input, can allow the assessment of an anomaly score which does not only give information on whether some data is anomalous but also how stark the anomaly is in comparison to existing data.

One definition of the anomaly detection setting [20] assumes that there exists a probability density function p_n from which normal data instances are generated:

$$X_n \sim p_n(x) = p(x|y=0) \tag{1}$$

with y being a label signaling that some data belongs to the normal class ($y = 0$) or abnormal class ($y = 1$). A dataset is usually composed of normal instances from X_n as well as anomalous instances from dataset X_a, with the latter being accordingly distributed with the anomalous distribution $X_a \sim p_a(x) = p(x|y=1)$. This results in a joint distribution containing both, normal and anomalous data points:

$$X_{total} \sim p_{total}(x) = (1-\Lambda)p_n(x) + \Lambda p_a(x) \tag{2}$$

where $\Lambda \in [0, 1]$ encodes the relative amount of anomalous points $p(y = 1)$. The task of anomaly detection then is to assess if data is drawn from the normal or abnormal data distribution.

2.2 Generative Adversarial Networks

Goodfellow et al. [21] introduce GANs as a framework consisting of two players, generator $\mathcal{G}$ and discriminator $\mathcal{D}$, following a minimax game with value function $V(\mathcal{G}, \mathcal{D})$. We call this the adversarial loss function $\mathcal{L}$:

$$\begin{aligned}\mathcal{L} &= \min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{D}, \mathcal{G}) \\ &= \mathbb{E}_{x\sim p_{data}(x)}\left[\log \mathcal{D}(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-\mathcal{D}(\mathcal{G}(z)))\right].\end{aligned} \tag{3}$$

In this adversarial game, the generator tries to create synthetic data similar to real data points $x \sim p_{data}, x \in \mathbb{R}^{d_x}$ with x consisting of n samples during training. In case of OC anomaly detection, all training samples belong to the normal class. The discriminator tries to distinguish data from the real training set from data which the generator $\mathcal{G}(z)$ synthetically generates using the latent noise vector $z \sim p_z$ as input. p_z often follows a Gaussian (commonly $z \sim \mathcal{N}(0, \mathrm{I})$) or uniform distribution with $z \in \mathbb{R}^{d_z}$ and $d_z \ll d_x$. The goal is to learn the parameters of $\mathcal{G}_\theta(z) \sim p_\theta$ s.t. p_θ is a potential candidate to represent p_{data}.

GANs have been especially useful on image data, and with CNNs being very effective on this domain, deep convolutional GANs (DCGANs) have been proposed by Radford et al. [22]. A central empirical result critical for many areas of research that have since evolved and which is most crucial for the detection of anomalies, is the existence of smooth transitions in latent space. This means that sampling similar noise z and using it as input for the generator also leads to the generation of similar images given sufficient training of the network.

This property has been observed on a variety of data structures: after proposed by DCGAN, this property was used for many tasks related to imaging which are based on this behavior, such as image generation and style transfer [23]. Furthermore, investigations on time series data using RNNs suggest similar behavior [24], [25] and Bojchevski et al. [26]

[1] We further publish an open source framework to evaluate various GAN-based anomaly detection approaches at `https://github.com/emundo/ecgan`.

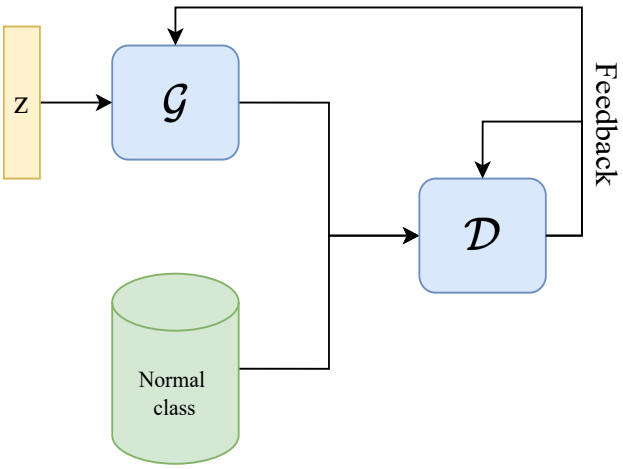


Figure 1: General training pipeline. The discriminator assesses the quality of generated and real data. This information is used to iteratively improve the discriminator as well as generator.

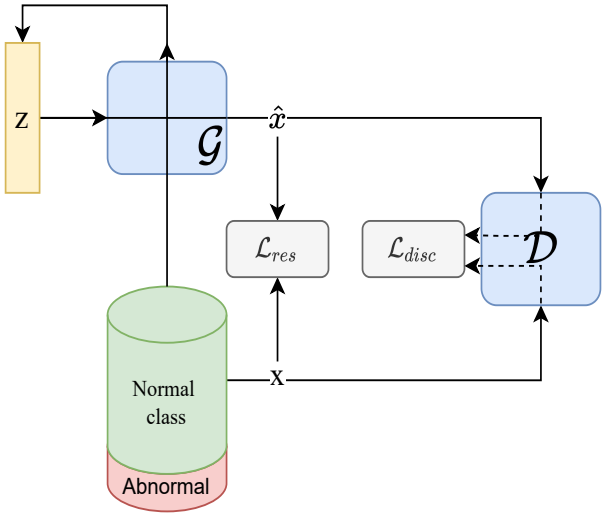


Figure 2: General anomaly detection pipeline. A given datum x is judged based on the discriminator as well as reconstruction loss between x and an arbitrarily similar sample $\hat{x}$. $\hat{x}$ can be retrieved by an explicit latent optimization minimizing the dissimilarity or implicitly by retrieving an inverse mapping using an encoding network.

visualize these transitions using attributes of graphs.

2.3 Anomaly Detection using Generative Adversarial Networks

The most widespread approach to GAN-based anomaly detection is based on learning the manifold of data of the normal class, meaning that the training data usually consists of only normal class data, i.e. $p_{data} \approx p_n$, such that $\Lambda = 0$ (Eq. 2) for the training data. The abnormal data is only used for validation and testing in the OC setting. It should be considered to include abnormal data during training as "synthetic" data in practical applications to improve the resulting learned generative procedure (e.g. using minimum likelihood regularization [27]) and to improve the separating hyperplane of the discriminator. Data points can be rated as normal or abnormal based on reconstructing the data x using $\mathcal{G}(z)$ (e.g. [28], [29]) and calculating a residual loss $\mathcal{L}_{res}$ using a well-defined and domain-dependent distance metric. Additionally, the discriminator can be used to assess if some datum belongs to the distribution represented by the generator by asserting a likelihood (e.g. [27]). A major approach in this area of research, AnoGAN, utilizes a combination of both components. The general training pipeline of GANs as well as the basic anomaly detection components have been visualized in Fig. 1 and Fig. 2. Before discussing AnoGAN and subsequently the most important developments in this domain and its applicability on a variety of data structures as well as practical or theoretical advances on this domain, we will briefly discuss existing applications and reviews of GAN-based anomaly detection.

Previous surveys on anomaly detection frequently focus on a broad array of methods (e.g. [30], [1]) or the general use of deep learning in anomaly detection and very general introductions to models or applications, e.g. Chalapathy et al. [31] for general deep learning based anomaly detection or Brophy et al. [32] for a more general use of GANs on time series data, which includes the use of anomaly detection. More recently, GAN-based anomaly detection has received a significant amount of attention and concurrent work exists which specifically reviews the usage of GANs for anomaly detection. Di Mattia et al. [9] perform a practical comparison of three major approaches in GAN-based anomaly detection. Sabuhi et al. [8] perform an exhaustive review of existing literature, investigating the domain of application, model architecture, datasets and evaluation metrics used by GAN-based anomaly detection. It shows a broad range of applications for GAN-based anomaly detection ranging from medical imaging [4] over the detection of deceptive reviews [33] to time series data [24] or videos [34]. While this work is an excellent resource for an overview of the currently used components, existing work rarely discusses the actual challenges of GAN-based anomaly detection apart from short remarks on training stability. Our work focuses on investigating these practical as well as theoretical obstacles which - to the best of our knowledge - have not been discussed in a systematic manner before.

3 The Fundamental Approach of AnoGAN

While GANs can be used to detect anomalies in a variety of ways, the procedure of AnoGAN by Schlegl et al. [4] can be considered central to most of these approaches and subsequent developments presented in this work are derived as adaptions from this approach. AnoGAN utilizes both, the difference in the features using the discriminator and its loss $\mathcal{L}_{disc}$, as well as the difference in data space, using the generator to calculate the absolute error as the residual loss $\mathcal{L}_{res}$. This is achieved by fully training a GAN on normal class data to learn the generator mapping $\mathcal{G} : z \mapsto x$. Dur-

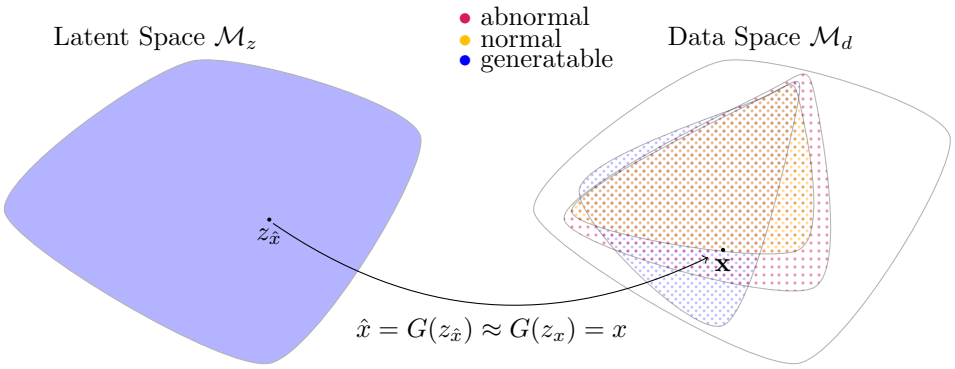


Figure 3: Simplified mapping between latent and data space. Each point in the lower dimensional latent space maps to a point in data space, resulting in a subspace of the data space which spans the data generatable by the GAN (blue area in $\mathcal{M}_d$). By training on only normal class data (yellow area in $\mathcal{M}_d$), it is attempted to restrict the generated data. In practice, the generatable data also includes abnormal data (red area in $\mathcal{M}_d$) and data which does not belong to the problem domain (e.g. if the task is to investigate CT images a valid image is generated but it is no realistic CT). Similarly, given a predefined threshold value for the discriminator error, one can separate a subset aiming to encompass normal class data as good as possible. By using a combination of both error components, we aim to minimize the amount of falsely classified samples.

ing inference, we try to determine if some novel datum x should be labeled as anomalous. To achieve this, it is required to find the noise vector z which is mapped as close as possible to x using the generator by interpolating through latent space: Initial noise z_1 is sampled randomly to generate $\mathcal{G}(z_1)$ in data space. The dissimilarity between x and $\mathcal{G}(z_1)$ is calculated. AnoGAN utilizes the absolute error but other common, well-defined distances have frequently been applied as proxies for the similarity of data points, including the euclidan distance. For time series data, RBF kernels have been a common choice and more time series specific measures such as dynamic time warping can easily be utilized. This distance is used to define a loss function to provide gradients that allow to move to some z_2 where $\mathcal{G}(z_2)$ is more similar to x than $\mathcal{G}(z_1)$. This is repeated Γ times to find the most similar image $\mathcal{G}(z_\Gamma)$ which can be constructed using the normal class manifold learned by the generator. Γ can either be a fixed value or be dynamically determined by a target similarity ϵ. A mixture of both strategies is commonly used where we try to find an ϵ-similar datum and interrupt optimization after a maximum of n_{max} steps in case the input is too dissimilar for the target similarity to be reached and to guarantee a maximum runtime. As soon as $\hat{x} = \mathcal{G}(z_\Gamma)$ is determined, it is compared to x using the similarity in data space by calculating the residual loss $\mathcal{L}_{res}$. In the case of images, and very similarly in the case of other regular data such as time series, $\mathcal{L}_{res}$ can be calculated by pointwise comparison of x and $\mathcal{G}(z_\Gamma)$:

$$\mathcal{L}_{res}(x, z_\Gamma) = |x - \mathcal{G}(z_\Gamma)|, \tag{4}$$

or another distance measure, depending on the target domain. It does not necessarily need to be the distance minimized during the latent optimization procedure, even though they do not differ in most applications. Afterwards, the discriminator is used to calculate the discriminative loss which enforces $\mathcal{G}(z_\Gamma)$ to lie on the learned manifold. Just as we try to force the generator to only produce healthy data given any valid z, the discriminator should only assign high confidence values to healthy data. The resulting discriminator loss is used by feeding $\mathcal{G}(z_\Gamma)$ to the discriminator, resulting in the following loss:

$$\mathcal{L}_{disc}(z_\Gamma, x) = \sigma(\mathcal{D}(\mathcal{G}(z_\Gamma), \alpha) \tag{5}$$

with σ being the sigmoid cross entropy which is used to describe the discriminator loss during training with logits $\mathcal{D}(\mathcal{G}(z_\Gamma))$ and targets $\alpha = 1$. The exact calculation of $\mathcal{L}_{disc}$ and $\mathcal{L}_{res}$ can differ: Schlegl et al. [4] further propose another $\mathcal{L}_{disc}$ based on feature matching

$$\mathcal{L}_{disc}(z_\Gamma, x) = |f(x) - f(G(z_\Gamma))|, \tag{6}$$

which has since been frequently applied [35], [29], [36]. The generator and discriminator are jointly used to calculate a combined loss which is a weighted sum of both components:

$$\mathcal{L}_{total}(z_\Gamma, x) = (1 - \lambda)\mathcal{L}_{res}(z_\Gamma) + \lambda\mathcal{L}_D(z_\Gamma). \tag{7}$$

Here, $\mathcal{L}_{total}(z_\Gamma, x)$ can be used directly to calculate an anomaly score. The anomaly score can be thresholded by some predefined or optimized τ to determine a label corresponding to x using $\mathcal{H} : \mathcal{L}_{total}(z_\Gamma, x) \mapsto \{0, 1\}$ with $\mathcal{H} = 0$ corresponding to normal samples and $\mathcal{H} = 1$ corresponding to abnormal samples respectively:

$$\mathcal{H}(\mathcal{L}_{total}(z_\Gamma, x), \tau) = \begin{cases} 0 \text{ if } \mathcal{L}_{total}(z_\Gamma, x) \leq \tau \\ 1 \text{ if } \mathcal{L}_{total}(z_\Gamma, x) > \tau \end{cases}. \tag{8}$$

Parts of the general anomaly detection procedure have been visualized in Fig. 3, the AnoGAN interpolation in Fig. 4. In practice, the sets are not convex and not easily interpolateable due to a complex loss surface when minimizing the dissimilarity.

4 Advances in GAN-based Anomaly Detection

We focus on five important challenges that should be considered when performing anomaly detection with GANs: speed and quality of the training process, restricting the latent

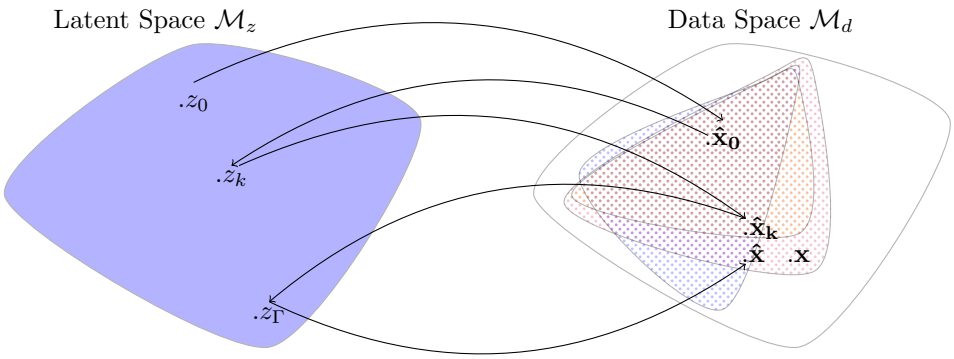


Figure 4: AnoGAN visualized. To approximate an abnormal sample x, initial latent vector z_0 is sampled and used to generate x_0. By minimizing the dissimilarity in data space, the latent vector z_Γ is retrieved, resulting in the best reconstruction $\hat{x}$ to x, i.e. minimizing the reconstruction error. The residual loss corresponds to the remaining difference in data space.

space, contaminated training data, the compositional choice of the anomaly score and inference performance. Some of these challenges introduce large and abstract areas of research and are related to each other. They can be split up further if desired, but shall act as a first reference point when considering anomaly detection with GANs or related generative models.

4.1 Speed and quality of the training process

Training GANs and the selection of suitable hyperparameters generally remains a non-trivial task: Choosing a sufficient latent distribution and dimensionality d_z ("the degree of compression" [37]) is not only crucial for inference time. It also has significant implications on the reconstruction of anomalous samples (e.g. [38]) and thus the reconstruction error and the anomaly detection performance in general. A small d_z can lead to non-convergence and insufficient preservation of information by the network because not all relevant features can be learned. If d_z is too large, training (and subsequently inference) are slowed down which is especially relevant if interpolation through latent space up until ϵ-similarity is chosen. Furthermore, too many irrelevant features might be learned, which hinders anomaly detection performance.

Due to the inherent instabilities of the originally proposed GAN architecture, which commonly causes mode collapse and further problems that hinder stable training, many researchers have resorted to practical workarounds. These workarounds include minibatch-discrimination [39] or using ensemble learning [27]. Since such methods often imply a significant overhead, novel objective functions such as the Wasserstein objective function (WGAN [40]) have found practical use. WGAN requires 1-Lipschitz continuity to stabilize training, which is enforced by weight clipping and has subsequently been replaced by ensuring that the norm of the gradient of the discriminator is penalized to stay (approximately) 1 in WGAN-GP [41]. This has also been successfully applied during training for anomaly detection, see e.g. [28], [35].

Depending on the underlying structure of data and networks used, different approaches to stabilize training can further be applied. These include spectral normalization [42], where a Lipschitz constraint to regularize the training is applied to the discriminator by normalizing all weights with the largest singular value, and a Lipschitz constraint to guarantee similar scores for data which is close in data space [43]. Progressive growing of GANs [44] attempts to stabilize and facilitate training by incrementally increasing the resolution of the data to iteratively increase the difficulty of the learned problem, which has since been used in the medical domain [45] as well as anomaly detection [46].

4.2 Restricting the latent space

While the previously presented approaches focus on the stability of the training process itself, the restriction of producing only normal class samples is less frequently discussed. In most settings, GANs are used to generalize and there is not always a need to completely restrict the latent space. While producing out-of-distribution data is often not only a byproduct but the explicit goal of image synthesis, it is difficult to restrict the generalizability of GANs to a subset of the generatable data. We initially assumed that the normal class data is a subset of $\mathbb{R}^{d_x}$ and $\mathcal{G} : z \mapsto x$ with $d_z \ll d_x$. Optimally, the latent space is a lower dimensional representation or approximation of the space of normal class data. However, it cannot be guaranteed that the generator is unable to produce abnormal class samples in the most common setting: Given $z \in \mathcal{N}(0, \mathrm{I})^{d_z}$, the sampled values during training will lie in a finite interval, likely with values only deviating up to a small number of standard deviations from zero in each dimension. Even though sampling large latent values is highly unlikely with $\lim_{z_i \to \infty} P(z_i) = 0 \; \forall \; \mathrm{i} \; \in \{1, .., d_z\}$, it is in principle possible to draw them since each value has a non-zero probability to be drawn from the (standard) Gaussian distribution [36]. Since AnoGAN only utilizes similarity based gradient information and disregards how likely the reconstruction is, it is possible to reconstruct highly unlikely data which has been observed by a variety of existing work [36], [47]. Tong et al. [43] discuss the sensitivity of the

reconstruction error in low density regions of normal class data and abnormal data close to the convex hull using autoencoders. They form the hypothesis that the model interpolates "too well for anomaly detection". To avoid this, we can restrict the possible values a priori, e.g. by using a uniform or truncated normal distribution, which introduces a tradeoff between variety and fidelity [48]. This has to the best of our knowledge not been applied to anomaly detection and requires a theoretical or empirical estimation of the cardinality of the problem domain. A different approach is to restrict the "accepted" reconstructions: given a multivariate standard Gaussian distributed latent space, it is possible to punish unlikely interpolations. This can be evaluated using the χ distribution where only certain deviations of the euclidean distance from origin of the latent vector, its latent norm, from its mode can be allowed by using predefined accepted standard deviations. This can also implicitly be learned if the latent norm is utilized as error component extending Eq. 7 using e.g. a grid search or SVM [36].

The ultimate goal of OC classification - and generation in this context - is to obtain a latent space where all instance from the latent space represent a datum from the given class [38]. With the approximation of the shape of the boundary separating in-class from out-of-class samples being the main goal, it becomes even more important to handle the data which is close to such boundary, i.e. the data close to the convex hull of normal data where at least $\mathcal{L}_{res}$ is low for anomalous points as well. Furthermore, $\mathcal{L}_{disc}$ is also low if not accounted for points close to the convex hull. This is increasingly important if we only interpolate through latent space: Since the generator might in general also produce abnormal data, moreso if d_z is chosen poorly, a bad initial random sampling might lead to some z_Γ producing $\mathcal{G}(z_\Gamma)$ which is far - the maximal allowed dissimilarity measured by ϵ - away from x, meaning that a normal point might be reconstructed with a high $\mathcal{L}_{res}$ or an abnormal point which could be reconstructed with a low $\mathcal{L}_{res}$ if ϵ is chosen poorly, leading to inaccurate classifications. This gets more important if the data is higher dimensional: Perera et al. [38] argue that especially complex data has a comparably weak novelty detection performance because the model automatically learns to represent some out-of-class objects if the shape is complex, leading to a low reconstruction error. This can further be problematic if the normality depends on the context, e.g. in the medical domain. This leads to a fuzzy and non-deterministic boundary since more information is required for a reliable detection. A stricter restriction of the manifold is thus required for many use cases. Sabokrou et al. [49] try to achieve a stricter decision boundary by enhancing inlier samples and distorting outliers, utilizing the addition of normal distributed noise. Koizumi et al. [50] generate a more suitable boundary by simulating non-normal data based on the Neyman-Pearson lemma, increasing the true positive rate by using an arbitrarily low false positive rate. They directly use an objective function which aims to increase the anomaly detection performance and utilize rejection sampling to simulate anomalous data and improve the resulting convex hull. Such a more distinctive boundary created by explicitly punishing the generation of abnormal data during training has received more attention recently (e.g. [46], [51]). Since the difficulties to establish the boundaries get larger the higher the dimensionality of the training data is, Liu et al. [52] utilize generative adversarial active learning to only generate informative potential outliers. Their approach shows to be especially effective on datasets with clusters of different shape as well as high ratios of irrelevant variables.

Further related approaches to divide anomalous from non-anomalous datasamples include Deep SVDDs [37] which have been used to learn a minimum-volume hypersphere in which the representations of, as close as possible to, all non-anomalous samples shall lie, regularizing the compactness of the learnt representation. Similarly, OCGAN [38] tackles the problem of restricting the latent representations by using a denoising autoencoder with a latent space which is forced to have bounded support enforced by the encoders output layer. Instead of only rewarding the generation of similar samples which might also be close to the convex hull, they add an adversarially trained discriminator in latent space to ensure that the representations of in-class examples resemble uniform random samples drawn from the same bounded space. The latent space is explored to produce potential out-of-class and high quality *informative-negative* examples which are fed into the network to steer the training to produce only in-class examples. Another approach which leverages the disciminator is to have a generator with the aim to produce weak anomalies to create such a distinct boundary [27]. Similarly, the use of a generator that does not attempt to mirror the true data distribution but improves low density areas to improve the generalization has been proposed [53]. Combining such approaches is also of interest includes using two generators with one generator learning the distribution of normal class data p_g^{normal} and one generator working as a *bad* generator learning p_g^{bad}. The discriminator receives data from p_{data}, p_g^{bad}, $p_g^{healthy}$, where data from p_g^{bad} would lead to an improved enforcement of the boundary between normal and abnormal data. Such a distinct boundary does not only have direct applications in common anomaly detection tasks: Neal et al. [54] use a GAN to generate counterfactual examples for unknown classes in image classification, reducing the influence of incorrect high-confidence predictions.

In a similar spirit to the previously mentioned investigation of the χ distribution mentioned before, Berg et al. [19] attempt to structure the latent space in such a way that the anomalous and normal samples are separated. The origin distance is used to measure the distance from the encoded image to the origin in the latent space, assuming that anomalous samples are farther away in latent space. It is possible that using the distance from a latent space centered around normality might allow a more distinctive mapping of points close to the convex hull than using only the reconstruction error.

4.3 Contaminated training data

Previously we have assumed that the data set contains correctly labeled instances. We would like to highlight an often overlooked, but in practice extremely important, problem: the contamination of data, meaning that most data sets rarely contain only correctly labeled data. The discussed generative models remain sensitive to outliers in training data and few anomalous points might contaminate the training set in OC classification [43]. One way to account for this is by assuming that a given sample (labeled as normal) can also come from an anomalous distribution [43] to allow a

margin of error.
Only very recently, researchers have begun to systematically investigate the impact of contamination, focusing on image data [19], [55], [46]. Some approaches are to reject potential anomalies during training [55] or to jointly train an encoder with the GAN in a progressive manner [19]. Salehi et al. [56] mention that the latent space may also primarily capture features that are shared by both, normal and anomalous data. Their work focuses on autoencoders but this holds true for GANs as well. They attempt to force their network to encapture features unique to the normal class using adversarial examples.

4.4 The compositional choice of the anomaly score

The multi-component anomaly score is one of the reasons why GANs are of interest for anomaly detection: Utilizing both, the reconstructive and the discriminative capabilities seems to increase the anomaly detection performance, most likely because different, complementary information is learned by the networks. In theory, either component should be sufficient to decide if some point is anomalous, with the reconstruction error allowing some limited explainability through visualization and it would be desirable to retrieve easy-to-interpret probabilities from the discriminator. In practice, it is usually impossible to train GANs to (near-) optimality and since the underlying approach is not unsupervised, acquiring and labeling the normal class data is seldom possible in the real world, especially without the before mentioned contamination. Exploring the practical differences and (dis-)advantages is important and current work rarely directly compares the performance of both: A variety of work argues that either the discriminator or the generator is partially unfit to deal with anomalous data without offering any experimental evidence. Most often, this is not specific to GANs, but focuses on the up- or downsides of the novelty likelihood, portrayed by the discrimination error in the GAN framework, or the residual error: On one hand, Deecke et al. [28] argue that the discrimination error is not equipped to deal with samples completely unlike the training data and only utilize the reconstruction error. On the other hand, it is argued that the reconstruction error does not always work very well in practice and suffers from a variety of problems such as intrinsic biases or instability if samples shall be reconstructed outside of the learned manifold [43], [57], [58], [51]. Pidhorskyi et al. [59] further argue that the reconstruction error only affects the noise portion of the model and does not include the signal portion.
Schlegl et al. [35] argue that only minimizing the reconstruction error is a subpar measure in regions of the latent space which are only sampled sparsely during training, but that the reconstruction error itself leads to better localization properties. Restricting the generator during training is an important body of future work (e.g. [56], [60], [61]).
Many problems presented here are shared with related frameworks such as (V)AEs, with An et al. [62] argueing that statistical anomaly detection methods, such as their proposed reconstruction probability, are a more objective, intuitive and robust anomaly score than the reconstruction error for VAEs. And while it is stated that they do "not require model specific thresholds for judging anomalies" using such probability based metrics, this is not necessarily true, even more so for GANs: using a fixed $\mathcal{G}$, the optimal discriminator is described by $\mathcal{D}^*_{\mathcal{G}} = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$ and in case of training up to optimality means $p_g = p_{data}$ (having a Jensen-Shannon divergence of zero) where the optimal discriminator will always return $\frac{1}{2}$ for all training samples as well as produced samples from the normal class manifold [21]. This has to be taken into account since it implies that the discrimination error might increase over time for both, normal class and good synthetic samples if a target value of 1 is used (Eq. 5). During optimization of the anomaly score, this has to be accounted for, e.g. using non-linear punishments of low discriminator predictions and frequent reparameterization of the anomaly score. Schlegl et al. [35] average the discriminator output across a high number of normal training samples and substract the discriminator output of some test data from the average discriminator score. While the problem solved by this methodology differs, it is one possible solution to the aforementioned problem. Some work tries to work around this by not using the direct discriminator score but the feature matching loss [4], [63], [36].
Choi et al. [64] further investigate the susceptiblity to out-of-distribution errors, argueing that likelihood models are in fact very susceptible to out-of-distribution samples, assigning large likelihoods to such samples (see also [65]). Even if the computation of the likelihood would be exact, a one-tailed test to check if some data has a low likelihood does not hold for high-dimensional data: They consider an isotropic high dimensional gaussian distribution, where a datum at the origin has maximum likelihood but is considered highly atypical because most of the probability mass lies in an annulus of radius $\sqrt{d_z}$. While likelihoods can determine whether a point lies in the support of a distribution, they do not reveal where the probability mass is concentrated. This also motivates the previously mentioned restriction of the latent space based on the χ distribution. Although the density estimation of GANs should not be able to account for probability mass, the generative ensemble presented in Choi et al. [64] demonstrate anomaly detection capabilities by combining density estimation and uncertainty estimation.
Berg et al. [19] compare the use of the distance in image space and the distance in latent space as a discriminative factor of reconstructed data. They find that the distance in image space (the reconstruction error as introduced above) is clearly preferable when it comes to a separation of the validation samples in latent space, and that a good distance in data space implies a good distance in latent space in practice but not vice versa.
Summarizing, the arguments supporting or opposing either error component can differ, but commonly named restrictions are the domain-specificity of the reconstruction error and the lack of its general interpretability or the black-box of the discriminator. Since both perspectives commonly argue with the instability of the opposite component for out-of-distribution samples, a systematic evaluation would be of great use. For now, it is likely and reasonable that both methods struggle with such data and the performance often depends on the problem domain. Restrictions on the latent space named in the previous section can help to reduce this impact and utilizing information of the latent space can be relevant for anomaly detection as well. Currently, the optimal weighting of the respective components should be evaluated for each experiment. Lüer et al. [36] find that the optimal weighting parameter between the error components further changes over time during training. They improve de-

tection performance by considering non-linear relationships between the error components using a non-linear SVM instead of a linear weighting of the anomaly score.

4.5 Inference speed and accuracy

One of the most significant practical limitation of the AnoGAN approach is that inference requires high amounts of computational resources and time. The required resources depend on a variety of variables: Deecke et al. [28] sample from multiple initial random z_0 to reduce the influence of erroneous local minima caused by initial sampling in unsuitable regions of the latent space. In general, this should increase the detection performance but also significantly increases the computational costs. The interpolation further depends on a variety of parameters, including the optimizer and its learning rate schedule, the similarity measure and resulting target similarity ϵ as well as the maximum allowed iterations n_{max} for the latent space interpolation. The computational costs are not necessarily always a restriction, e.g. inference time is often less relevant if medical images are investigated. But many real-world use cases involve constant monitoring and thus evaluations of data which is changing at a fast pace. An example is time series analysis in intrusion detection or medical monitoring, where the non-trivial and non-convex optimization is too expensive. Additional to the inference speed, the optimizer, ϵ and the allowed interpolation iterations are strongly dependent of each other and strongly influence the anomaly detection performance: Each use case requires the selection of such parameters and a low ϵ or high amount of interpolation iterations might lead to the generation of anomalous samples while a high ϵ or low amount of allowed interpolation iterations leads to dissimilar samples which do not correspond to the true anomalousness. To avoid the costly and error prone parameter selection and optimization, one can learn an additional, inverse mapping to $\mathcal{G}$, $E(x) = \mathcal{G}^{-1}(x) = z$. Following prior work, especially adversarial learned inference [66] and adversarial feature learning [67], Zenati et al. [68] utilize a bidirectional GAN to learn such a mapping for the task of anomaly detection. The discriminator does not only use $\mathcal{G}(z)$ or x as input but $(E(x), x)$ or $(z, \mathcal{G}(z))$. The mapping of the encoder can either be learned jointly during training [68], [69] but also after training [35]. Berg et al. [19] report that the joint training led to a better separation of normal and anomalous samples in latent space.

The work is closely related to CycleGAN [70], which utilize a cycle consistency loss to obtain such an inverse mapping which is furthermore cycle consistent: While a mapping from a bidirectional GAN learns *some* mapping $E : x \to z$, the inverse mapping of CycleGAN enforces that $\mathcal{G}$ and E are inverses or inverse approximations *of each other*. This means that both mappings are bijections, which is also desirable for the task of consistent anomaly detection, i.e. $E(\mathcal{G}(z)) \approx z$ and $\mathcal{G}(E(x)) \approx x$ corresponding to the forward and backward cycle-consistency respectively. This has since been adopted for anomaly detection and leads to a significant speedup during inference [63], [71], [72].

Similarly to the BiGAN approach, a significant body of work has evolved around the use of adversarially trained autoencoders. While using a (variational) autoencoder deviates from the likelihood-free principle of traditional GANs, training and mode coverage are significantly improved. The extension of using the discriminator additionally to the autoencoder has shown to improve training as well as detection [29], [36], [73]. Here, the reconstruction error (Eq. 4) as well as the discriminator error, frequently using the feature matching error from Eq. 6, remain core components. Additionally, the latent error has been utilized to leverage properties of the latent space [36] or more generally to ensure learning to correctly encode normal class data. One notable example is GANomaly [69] which utilizes a second encoder to retrieve a latent representation of the generated image and learns to minimize the difference of the parametrizations of both encoders. Using such mappings to latent space, the model is not as dependent on random initial noise anymore, but subsequently depends even more on a sufficient inverse mapping, which can be difficult to learn by itself. Inverse mappings have since been widely adopted, not showing any systematic deficits in the performance and reporting a significant speedup during inference, see Table 1 (evaluated on the beatwise preprocessed MITBIH dataset [74] using a β-VAEGAN[36]). Latent optimization is based on the similarity measured by an RBF-Kernel optimized using Adam [75] with an adapting learning. The (runtime as well as anomaly detection) performance largely depends on the previously mentioned parameters, further including the batch size used during optimization since similarity is frequently calculated batch-wise. However, this averaged similarity can distort the results and should be avoided by evaluating the similarity per sample. Comparability across datasets and across variations of these parameters is very limited. Due to only utilizing the forward pass of the encoder-based architecture, higher batch sizes can lead to significantly faster inference while retaining equally good detection performance, improving inference time from 1.9 ms on GPU (2.4 ms on CPU) using a batch size of 1 to 0.08 ms on GPU using a batch size of 512.

Table 1: Inference times (consumer GPU) for a β-VAEGAN model using different anomaly detection approaches on 22427 test samples of the beatwise preprocessed MITBIH dataset.

	ϵ	n_{max}	batch size	**Runtime (ms)**
`AnoGAN`	0.005	500	1	1139.7 ± 483.3
`AnoGAN`	0.005	500	64	38.27 ± 2.7
`AnoGAN`	0.005	1000	1	2254.4 ± 997.6
`AnoGAN`	0.05	500	1	325.8 ± 560.5
`AnoGAN`	0.05	100	1	77.9 ± 117.7
`Encoder`	-	-	1	$\mathbf{1.9 \pm 0.3}$

5 Discussion

We identify five major, intertwined obstacles GAN-based anomaly detection needs to tackle: speed and quality of the training process, restrictions to the latent space, contaminated training data, novel anomaly score components and compositions as well as inference accuracy and speed. The **speed and quality of the training process** is a fundamental requirement to make GAN-based anomaly detection suitable for a wide array of applications. Adversarial generative models generalize well, which is not always a desired property during anomaly detection and requires **restrictions of the latent space** to produce only normal class

data. In general, losses are usually weighted

$$\mathcal{L}_{total} = \sum_{i=1}^{k} \mathcal{L}_i \cdot \lambda_i, \quad \sum_{i=1}^{k} \lambda_i = 1, \tag{9}$$

with $k = 1$ [28] or $k = 2$ [4] being common choices. But in general, **novel anomaly score components and compositions** such as explicit information about the latent space and non-linear weigthings can be incorporated. The performance and speed of AnoGAN depends of various parameters which are relevant for the search of $\mathcal{G}(z_\Gamma)$ - the optimizer, target dissimilarity ϵ and the maximum amount of iterations, as well as the component weighting λ and the anomaly score threshold τ. While increasing the amount of components can improve the performance, it also increases the time for optimizing λ_i and τ, making a large search space computational infeasible. The optimization process can be performed by a variety of mechanisms, including a naive grid search or by optimizing an SVM [36]. Using fixed weighting parameters and anomaly score thresholds can hinder performance and should not be utilized. The optimization is usually still cheap in terms of computational resources required in comparison to the GAN training, but one should be aware of the biases the optimization might introduce.

To avoid tuning interpolation parameters and **speed up inference and accuracy**, inverse mappings have been utilized in a variety of settings. Lastly, OC training still requires a definition and selection of normal class data, which can suffer from **contaminated data**. Additionally, more extensive research for non-image data will be required: Time series are usually split up into subsequences during training as well as evaluation. MAD-GAN [76] report more false positives for larger subsequence lengths. A possible explanation is that more training data is required to model larger time series due to the curse of dimensionality. Furthermore, the subsequence length likely influences the (in)stability of the training process.

Although a variety of challenges remain open and while a significant amount of progress will still be the crucial requirement for some practical applications, the recent success and interest in GANs on the domain of anomaly detection sketches their potential. Existing work focuses on the medical domain [8], which especially benefits from the GAN-based procedure: By only requiring the definition of a normal class, it is possible to detect unknown anomalies/pathologies which can be investigated in-depth. However, guarantees regarding the detection performance are crucial for many medical use cases and the problem of generating abnormal data in the one-class setting has not yet received sufficient attention.

More investigations into the comparison of the learned structure between GANs and similarly used representation learning networks (especially autoencoders) are of interest to improve understanding and ultimately performance of existing work. First attempts on comparing generative models with traditional models (e.g. [77]) commonly do not cover the pecularities of the respective models sufficiently, especially the sensitivity to hyperparameters. Automated machine learning approaches might allow a more robust and fair comparison.

GANs are most commonly used if the underlying dataset is heavily imbalanced, i.e. if anomalies are scarce and obtaining extensive data of the normal class is feasible. Most existing work focuses on the comparison of a narrow set of metrics, most commonly F_β and AUC. The F_β score is especially susceptible to imbalanced data and since the test data often also includes only few anomalies, the F_β score is less meaningful with the resulting score usually being very high and misrepresenting the actual capabilities. Using more balanced measures, such as be the phi coefficient, might improve the insights that can be gained.

This work has focused on GAN-based approaches that are explicitly used to classify data, usually as a feature extractor or via reconstructions, not implicitly e.g. through generating data to augment other anomaly detection methods or related tasks such as segmentation [78] or data imputation [79]. Learning the normal class manifold and using GANs to derive or detect changes from it is not limited to anomaly detection in a binary classification setting. It can also be extended to multiclass anomaly detection and sufficient learning of the class boundaries also allows to generate more specific data. In medicine, such data can be used for augmentation [80], [78], [81], [82], [83], [80] or practical training of medical professionals [84]. This has since allowed to improve anomaly detection performance on tasks with only few available training samples. More extensive evaluations on the implications and possible fallacies are still required. Approaches to protect the privacy of the patients (such as differential privacy, e.g. [85]), can be of high importance in this case and need to be incorporated in advance. Augmentations can also be of relevance during anomaly detection since it might be useful to use generative models to oversample infrequent normal samples (e.g. [86]) which are often close to the convex hull and are commonly misclassified as false positives.

GAN-based anomaly detection relies on significant amounts of normal class data. Since no negative information is incorporated in most settings, the detection of anomalies has proven to be difficult if the similarity in data space is high. If only the total anomaly score is available, it is more difficult to interpret the results, even though visualizations have started to give significant insights into the structure. Furthermore, training GANs is a significantly more ambitious, complex and time-consuming approach than solely utilizing a discriminative model. The training procedure requires a significant amount of domain knowledge and has a higher-dimensional hyperparameter space in comparison to many more traditional methods, not necessarily allowing the fast training of a reasonably good baseline. However, the use of implicit generative models can help improve generalizability on complex data manifolds, partially reducing the assumptions that have to be posed to define and detect anomalies. Implicit generative modeling thrives if it is difficult to explicitly describe anomalies, which is especially common for high dimensional data. GANs provide a general framework which can be applied to many different data structures and established components can be easily incorporated (e.g. the use of CNNs for image data). Another advantage of more recent methods is the speed of inference in comparison to many related OC anomaly detection methods. An additional benefit of using GANs is the reduced amount of data required in a semi-supervised learning process in comparison to similar fully supervised approaches which is especially relevant in domains where labeling is expensive [87].

Independent of the actual task, many of the listed challenges are applicable to other generative models. This in-

cludes (also variational or adversarial) autoencoders which are used in a similar way to detect anomalies, e.g. [88] or [51]. In-depth comparisons are left for future investigation. The similarities of their structure and the compression of data into a latent representation of a lower dimension than the data space and the use of the encoder, which is commonly used to calculate a residual/reconstruction error, are of high relevance. VAEs themselves suffer from the problem of blurriness in image space. This most likely results from the diffuse probability mass distribution over the data space due to the combination of the conditional independence assumption with the maximum likelihood training paradigm [66], [89]. Due to the comparably new state of inverse mapping, possible disadvantages in quality or learned structure have yet to be explored on a larger scale.

6 Concluding Remarks

Pairing generative models and their reconstructive capabilities with adversarial feedback has shown state-of-the-art performance on many complex and high dimensional problems, especially if data is heavily imbalanced towards one class. The generative procedure still suffers from unstable training, frequently leading to deviations from implicit generative modeling by combining autoencoders with an adversarial component. The addition of an inverse mapping from data space to latent space has further significantly improved inference times. While the achieved performances are remarkable, sensitive applications such as medical usecases - currently the predominant domain GAN-based anomaly detection is applied to [8] - still have to be treated with caution when utilized in practice. Apart from improving the efficiency and effectiveness of models themselves, future work especially needs to account for properties of the latent space and the possibility to generate data which is either abnormal or not belonging to the problem domain at all. This also includes work on utilizing negative information to strengthen the decision boundary. The influence of GAN-generated augmentation data on arbitrary anomaly detection approaches requires further investigations, especially regarding their theoretical limitations. Weightings of the individual anomaly detection components should be performed for each experiment and can significantly differ between models and datasets. Even though the applicability largely depends on the availability of normal class data, GANs can be very versatile and the trained GAN model can be used to perform a large variety of auxiliary tasks, making it a very interesting modelling approach. Many of our findings are also applicable more generally on other classification tasks using different generative models, such as diffusion models.

Acknowledgements

This work is supported by the Bavarian Research Foundation under grant AZ-1419-20.

7 REFERENCES

[1] Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41**(3) (2009) 1–58

[2] Denning, D.E.: An intrusion-detection model. IEEE Transactions on software engineering (2) (1987) 222–232

[3] Das, M., Parthasarathy, S.: Anomaly detection and spatio-temporal analysis of global climate system. In: Proceedings of the third international workshop on knowledge discovery from sensor data. (2009) 142–150

[4] Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U., Langs, G.: Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: International conference on information processing in medical imaging, Springer (2017) 146–157

[5] Abdallah, A., Maarof, M.A., Zainal, A.: Fraud detection system: A survey. Journal of Network and Computer Applications **68** (2016) 90–113

[6] Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: International Conference on Machine Learning, PMLR (2017) 1321–1330

[7] Malhotra, P., Vig, L., Shroff, G., Agarwal, P.: Long short term memory networks for anomaly detection in time series. In: Proceedings. Volume 89., Presses universitaires de Louvain (2015)

[8] Sabuhi, M., Zhou, M., Bezemer, C.P., Musilek, P.: Applications of generative adversarial networks in anomaly detection: A systematic literature review. IEEE Access (2021)

[9] Di Mattia, F., Galeone, P., De Simoni, M., Ghelfi, E.: A survey on gans for anomaly detection. arXiv preprint arXiv:1906.11632 (2019)

[10] Hawkins, D.M.: Identification of outliers. Volume 11. Springer (1980)

[11] Akkus, Z., Galimzianova, A., Hoogi, A., Rubin, D.L., Erickson, B.J.: Deep learning for brain mri segmentation: state of the art and future directions. Journal of digital imaging **30**(4) (2017) 449–459

[12] Liu, F., Jang, H., Kijowski, R., Bradshaw, T., McMillan, A.B.: Deep learning mr imaging–based attenuation correction for pet/mr imaging. Radiology **286**(2) (2018) 676–684

[13] Ding, Y., Sohn, J.H., Kawczynski, M.G., Trivedi, H., Harnish, R., Jenkins, N.W., Lituiev, D., Copeland, T.P., Aboian, M.S., Mari Aparici, C., et al.: A deep learning model to predict a diagnosis of alzheimer disease by using 18f-fdg pet of the brain. Radiology **290**(2) (2019) 456–464

[14] Hannun, A.Y., Rajpurkar, P., Haghpanahi, M., Tison, G.H., Bourn, C., Turakhia, M.P., Ng, A.Y.: Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. Nature medicine **25**(1) (2019) 65

[15] Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data mining and knowledge discovery **29**(3) (2015) 626–688

[16] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)

[17] Abati, D., Porrello, A., Calderara, S., Cucchiara, R.: Latent space autoregression for novelty detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 481–490

[18] Yeung, D.Y., Chow, C.: Parzen-window network intrusion detectors. In: Object recognition supported by user interaction for service robots. Volume 4., IEEE (2002) 385–388

[19] Berg, A., Ahlberg, J., Felsberg, M.: Unsupervised learning of anomaly detection from contaminated image data using simultaneous encoder training. arXiv preprint arXiv:1905.11034 (2019)

[20] Pimentel, T., Monteiro, M., Viana, J., Veloso, A., Ziviani, N.: A generalized active learning approach for unsupervised anomaly detection. stat **1050** (2018) 23

[21] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. (2014) 2672–2680

[22] Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)

[23] Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 4401–4410

[24] Li, D., Chen, D., Jin, B., Shi, L., Goh, J., Ng, S.K.: Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In: International Conference on Artificial Neural Networks, Springer (2019) 703–716

[25] Lüer, F., Mautz, D., Böhm, C.: Anomaly detection in time series using generative adversarial networks. In: 2019 International Conference on Data Mining Workshops (ICDMW), IEEE (2019) 1047–1048

[26] Bojchevski, A., Shchur, O., Zügner, D., Günnemann, S.: Netgan: Generating graphs via random walks. arXiv preprint arXiv:1803.00816 (2018)

[27] Wang, C., Zhang, Y.M., Liu, C.L.: Anomaly detection via minimum likelihood generative adversarial networks. In: 2018 24th International Conference on Pattern Recognition (ICPR), IEEE (2018) 1121–1126

[28] Deecke, L., Vandermeulen, R., Ruff, L., Mandt, S., Kloft, M.: Image anomaly detection with generative adversarial networks. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer (2018) 3–17

[29] Zhou, B., Liu, S., Hooi, B., Cheng, X., Ye, J.: Beatgan: Anomalous rhythm detection using adversarially generated time series. In: IJCAI. (2019) 4433–4439

[30] Pimentel, M.A., Clifton, D.A., Clifton, L., Tarassenko, L.: A review of novelty detection. Signal Processing **99** (2014) 215–249

[31] Chalapathy, R., Chawla, S.: Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407 (2019)

[32] Brophy, E., Wang, Z., She, Q., Ward, T.: Generative adversarial networks in time series: A survey and taxonomy. arXiv preprint arXiv:2107.11098 (2021)

[33] Aghakhani, H., Machiry, A., Nilizadeh, S., Kruegel, C., Vigna, G.: Detecting deceptive reviews using generative adversarial networks. In: 2018 IEEE Security and Privacy Workshops (SPW), IEEE (2018) 89–95

[34] Ravanbakhsh, M., Nabi, M., Sangineto, E., Marcenaro, L., Regazzoni, C., Sebe, N.: Abnormal event detection in videos using generative adversarial nets. In: 2017 IEEE International Conference on Image Processing (ICIP), IEEE (2017) 1577–1581

[35] Schlegl, T., Seeböck, P., Waldstein, S.M., Langs, G., Schmidt-Erfurth, U.: f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. Medical image analysis **54** (2019) 30–44

[36] Lüer, F., Dolgich, M., Weber, T., Böhm, C.: Adversarial anomaly detection using gaussian priors and nonlinear anomaly scores. In: 2023 International Conference on Data Mining Workshops (ICDMW), IEEE (2023)

[37] Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S.A., Binder, A., Müller, E., Kloft, M.: Deep one-class classification. In: International conference on machine learning. (2018) 4393–4402

[38] Perera, P., Nallapati, R., Xiang, B.: Ocgan: One-class novelty detection using gans with constrained latent representations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 2898–2906

[39] Salimans, T., Goodfellow, I.J., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R., eds.: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. (2016) 2226–2234

[40] Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint arXiv:1701.07875 (2017)

[41] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: Advances in neural information processing systems. (2017) 5767–5777

[42] Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net (2018)

[43] Tong, A., Wolf, G., Krishnaswamy, S.: A lipschitz-constrained anomaly discriminator framework. arXiv preprint arXiv:1905.10710 (2019)

[44] Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)

[45] Baur, C., Albarqouni, S., Navab, N.: Generating highly realistic images of skin lesions with gans. In: OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis. Springer (2018) 260–267

[46] Kimura, M., Yanagihara, T.: Anomaly detection using gans for visual inspection in noisy training data. In: Asian Conference on Computer Vision, Springer (2018) 373–385

[47] Yoon, S., Noh, Y.K., Park, F.C.: Autoencoding under normalization constraints. arXiv preprint arXiv:2105.05735 (2021)

[48] Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)

[49] Sabokrou, M., Khalooei, M., Fathy, M., Adeli, E.: Adversarially learned one-class classifier for novelty detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 3379–3388

[50] Koizumi, Y., Saito, S., Uematsu, H., Kawachi, Y., Harada, N.: Unsupervised detection of anomalous sound based on deep learning and the neyman–pearson lemma. IEEE/ACM Transactions on Audio, Speech, and Language Processing **27**(1) (2018) 212–224

[51] Kimura, D., Chaudhury, S., Narita, M., Munawar, A., Tachibana, R.: Adversarial discriminative attention for robust anomaly detection. In: The IEEE Winter Conference on Applications of Computer Vision. (2020) 2172–2181

[52] Liu, Y., Li, Z., Zhou, C., Jiang, Y., Sun, J., Wang, M., He, X.: Generative adversarial active learning for unsupervised outlier detection. IEEE Transactions on Knowledge and Data Engineering (2019)

[53] Dai, Z., Yang, Z., Yang, F., Cohen, W.W., Salakhutdinov, R.R.: Good semi-supervised learning that requires a bad gan. In: Advances in neural information processing systems. (2017) 6510–6520

[54] Neal, L., Olson, M., Fern, X., Wong, W.K., Li, F.: Open set learning with counterfactual images. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 613–628

[55] Beggel, L., Pfeiffer, M., Bischl, B.: Robust anomaly detection in images using adversarial autoencoders. arXiv preprint arXiv:1901.06355 (2019)

[56] Salehi, M., Arya, A., Pajoum, B., Otoofi, M., Shaeiri, A., Rohban, M.H., Rabiee, H.R.: Arae: Adversarially robust training of autoencoders improves novelty detection. Neural Networks **144** (2021) 726–736

[57] Šmídl, V., Bím, J., Pevnỳ, T.: Anomaly scores for generative models. arXiv preprint arXiv:1905.11890 (2019)

[58] An, J., Cho, S.: Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE **2**(1) (2015) 1–18

[59] Pidhorskyi, S., Almohsen, R., Doretto, G.: Generative probabilistic novelty detection with adversarial autoencoders. In: Advances in neural information processing systems. (2018) 6822–6833

[60] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., Madry, A.: Adversarial examples are not bugs, they are features. In: Advances in Neural Information Processing Systems. (2019) 125–136

[61] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)

[62] An, J., Cho, S.: Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE **2**(1) (2015) 1–18

[63] Zenati, H., Romain, M., Foo, C.S., Lecouat, B., Chandrasekhar, V.: Adversarially learned anomaly detection. In: 2018 IEEE International Conference on Data Mining (ICDM), IEEE (2018) 727–736

[64] Choi, H., Jang, E., Alemi, A.A.: Waic, but why? generative ensembles for robust anomaly detection. arXiv preprint arXiv:1810.01392 (2018)

[65] Nalisnick, E., Matsukawa, A., Teh, Y.W., Gorur, D., Lakshminarayanan, B.: Do deep generative models know what they don't know? arXiv preprint arXiv:1810.09136 (2018)

[66] Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., Courville, A.: Adversarially learned inference. arXiv preprint arXiv:1606.00704 (2016)

[67] Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. arXiv preprint arXiv:1605.09782 (2016)

[68] Zenati, H., Foo, C.S., Lecouat, B., Manek, G., Chandrasekhar, V.R.: Efficient gan-based anomaly detection. arXiv preprint arXiv:1802.06222 (2018)

[69] Akcay, S., Atapour-Abarghouei, A., Breckon, T.P.: Ganomaly: Semi-supervised anomaly detection via adversarial training. In: Asian conference on computer vision, Springer (2018) 622–637

[70] Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision. (2017) 2223–2232

[71] Hirose, N., Sadeghian, A., Vázquez, M., Goebel, P., Savarese, S.: Gonet: A semi-supervised deep learning approach for traversability estimation. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE (2018) 3044–3051

[72] Armanious, K., Jiang, C., Abdulatif, S., Küstner, T., Gatidis, S., Yang, B.: Unsupervised medical image translation using cycle-medgan. In: 2019 27th European Signal Processing Conference (EUSIPCO), IEEE (2019) 1–5

[73] van Hespen, K.M., Zwanenburg, J.J., Dankbaar, J.W., Geerlings, M.I., Hendrikse, J., Kuijf, H.J.: An anomaly detection approach to identify chronic brain infarcts on mri. Scientific Reports **11**(1) (2021) 1–10

[74] Kachuee, M., Fazeli, S., Sarrafzadeh, M.: Ecg heartbeat classification: A deep transferable representation. In: 2018 IEEE international conference on healthcare informatics (ICHI), IEEE (2018) 443–444

[75] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[76] Li, D., Chen, D., Jin, B., Shi, L., Goh, J., Ng, S.K.: Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In: International Conference on Artificial Neural Networks, Springer (2019) 703–716

[77] Škvára, V., Pevnỳ, T., Šmídl, V.: Are generative deep models for novelty detection truly better? arXiv preprint arXiv:1807.05027 (2018)

[78] Mahmood, F., Borders, D., Chen, R., McKay, G.N., Salimian, K.J., Baras, A., Durr, N.J.: Deep adversarial training for multi-organ nuclei segmentation in histopathology images. IEEE transactions on medical imaging (2019)

[79] Luo, Y., Cai, X., Zhang, Y., Xu, J., et al.: Multivariate time series imputation with generative adversarial networks. In: Advances in Neural Information Processing Systems. (2018) 1596–1607

[80] Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., Greenspan, H.: Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. Neurocomputing **321** (2018) 321–331

[81] Bowles, C., Chen, L., Guerrero, R., Bentley, P., Gunn, R., Hammers, A., Dickie, D.A., Hernández, M.V., Wardlaw, J., Rueckert, D.: Gan augmentation: Augmenting training data using generative adversarial networks. arXiv preprint arXiv:1810.10863 (2018)

[82] Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., Greenspan, H.: Synthetic data augmentation using gan for improved liver lesion classification. In: 2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018), IEEE (2018) 289–293

[83] Han, C., Murao, K., Noguchi, T., Kawata, Y., Uchiyama, F., Rundo, L., Nakayama, H., Satoh, S.: Learning more with less: Conditional pggan-based data augmentation for brain metastases detection using highly-rough annotation on mr images. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. (2019) 119–127

[84] Han, C., Hayashi, H., Rundo, L., Araki, R., Shimoda, W., Muramatsu, S., Furukawa, Y., Mauri, G., Nakayama, H.: Gan-based synthetic brain mr image generation. In: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), IEEE (2018) 734–738

[85] Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. (2016) 308–318

[86] Lim, S.K., Loo, Y., Tran, N.T., Cheung, N.M., Roig, G., Elovici, Y.: Doping: Generative data augmentation for unsupervised anomaly detection with gan. In: 2018 IEEE International Conference on Data Mining (ICDM), IEEE (2018) 1122–1127

[87] Madani, A., Moradi, M., Karargyris, A., Syeda-Mahmood, T.: Semi-supervised learning with generative adversarial networks for chest x-ray classification with ability of data domain adaptation. In: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), IEEE (2018) 1038–1042

[88] Chen, X., Konukoglu, E.: Unsupervised detection of lesions in brain mri using constrained adversarial auto-encoders. arXiv preprint arXiv:1806.04972 (2018)

[89] Theis, L., Oord, A.v.d., Bethge, M.: A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844 (2015)

Exploring the Potential of Large Language Models (LLMs) in Learning on Graphs

Zhikai Chen[1], Haitao Mao[1], Hang Li[1], Wei Jin[3], Hongzhi Wen[1],
Xiaochi Wei[2], Shuaiqiang Wang[2], Dawei Yin[2]
Wenqi Fan[4], Hui Liu[1], Jiliang Tang[1]
[1]Michigan State University [2] Baidu Inc. [3] Emory University
[4] The Hong Kong Polytechnic University
{chenzh85, haitaoma, lihang4, wenhongz, liuhui7, tangjili}@msu.edu,
{weixiaochi, wangshuaiqiang}@baidu.com, yindawei@acm.org,
wei.jin@emory.edu,
wenqifan03@gmail.com

ABSTRACT

Learning on Graphs has attracted immense attention due to its wide real-world applications. The most popular pipeline for learning on graphs with textual node attributes primarily relies on Graph Neural Networks (GNNs), and utilizes shallow text embedding as initial node representations, which has limitations in general knowledge and profound semantic understanding. In recent years, Large Language Models (LLMs) have been proven to possess extensive common knowledge and powerful semantic comprehension abilities that have revolutionized existing workflows to handle text data. In this paper, we aim to explore the potential of LLMs in graph machine learning, especially the node classification task, and investigate two possible pipelines: *LLMs-as-Enhancers* and *LLMs-as-Predictors*. The former leverages LLMs to enhance nodes' text attributes with their massive knowledge and then generate predictions through GNNs. The latter attempts to directly employ LLMs as standalone predictors. We conduct comprehensive and systematical studies on these two pipelines under various settings. From comprehensive empirical results, we make original observations and find new insights that open new possibilities and suggest promising directions to leverage LLMs for learning on graphs. Our codes and datasets are available at: `https://github.com/CurryTang/Graph-LLM` .

1. INTRODUCTION

Graphs are ubiquitous in various disciplines and applications, encompassing a wide range of real-world scenarios [73]. Many of these graphs have nodes that are associated with text attributes, resulting in the emergence of text-attributed graphs, such as citation graphs [23; 57] and product graphs [5]. For example, in the OGBN-PRODUCTS dataset [23], each node represents a product, and its corresponding textual description is treated as the node's attribute. These graphs have seen widespread use across a myriad of domains, from social network analysis [31], information retrieval [86], to a diverse range of natural language processing tasks [37; 76].

Given the prevalence of text-attributed graphs (TAGs), we aim to explore how to effectively handle these graphs, with a focus on the node classification task. Intuitively, TAGs provide both node attribute and graph structural information. Thus, it is important to effectively capture both while modeling their interrelated correlation. Graph Neural Networks (GNNs) [38] have emerged as the de facto technique for handling graph-structured data, often leveraging a message-passing paradigm to effectively capture the graph structure. To encode textual information, conventional pipelines typically make use of non-contextualized shallow embeddings e.g., Bag-of-Words [20] and Word2Vec [42] embeddings, as seen in the common graph benchmark datasets [23; 57], where GNNs are subsequently employed to process these embeddings. Recent studies demonstrate that these non-contextualized shallow embeddings suffer from some limitations, such as the inability to capture polysemous words [51] and deficiency in semantic information [41; 12], which may lead to sub-optimal performance on downstream tasks.

Compared to these non-contextualized shallow textual embeddings, large language models (LLMs) present massive context-aware knowledge and superior semantic comprehension capability through the process of pre-training on large-scale text corpora [48; 12]. This knowledge achieved from pre-training has led to a surge of revolutions for downstream NLP tasks [85]. Exemplars such as ChatGPT and GPT4 [46], equipped with hundreds of billions of parameters, exhibit superior performance [2] on numerous text-related tasks from various domains. Considering the exceptional ability of these LLMs to process and understand textual data, a pertinent question arises: (1) *Can we leverage the knowledge of LLMs to compensate for the deficiency of contextualized knowledge and semantic comprehension inherent in the conventional GNN pipelines?* In addition to the knowledge learned via pre-training, recent studies suggest that LLMs present preliminary success on tasks with implicit graph structures such as recommendation [35; 14], ranking [26], and multi-hop reasoning [7], in which LLMs are adopted to make the final predictions. Given such success, we further question: (2) *Can LLMs, beyond merely integrating with GNNs, independently perform predictive tasks with explicit graph structures?* In this paper, we aim to embark upon a preliminary investigation of these two questions by undertaking a series of extensive empirical analyses. Particularly, the key challenge is how to design an LLM-compatible pipeline for graph learn-

ing tasks. Consequently, we explore two potential pipelines to incorporate LLMs: (1) *LLMs-as-Enhancers*: LLMs are adopted to enhance the textual information; subsequently, GNNs utilize refined textual data to generate predictions. (2) *LLMs-as-Predictors*: LLMs are adapted to generate the final predictions, where structural and attribute information is present completely through natural languages.

In this work, we embrace the challenges and opportunities to study the utilization of LLMs in graph-related problems and aim to deepen our understanding of *the potential of LLMs on graph machine learning*, with a focus on the node classification task. **First**, we aim to investigate how LLMs can enhance GNNs by leveraging their extensive knowledge and semantic comprehension capability. It is evident that different types of LLMs possess varying levels of capability, and more powerful models often come with more usage restrictions [59; 85; 51]. Therefore, we strive to design different strategies tailored to different types of models, and better leverage their capabilities within the constraints of these usage limitations. **Second**, we want to explore how LLMs can be adapted to explicit graph structures as a predictor. A principal challenge lies in crafting a prompt that enables the LLMs to effectively use structural and attribute information. To address this challenge, we attempt to explore what information can assist LLMs in better understanding and utilizing graph structures. Through these investigations, we make some insightful observations and gain a better understanding of the capabilities of LLMs in graph machine learning.

Contributions. Our contributions are summarized as follows:

1. We explore two pipelines that incorporate LLMs to handle TAGs: *LLMs-as-Enhancers* and *LLMs-as-Predictors*. The first pipeline treats the LLMs as attribute enhancers, seamlessly integrating them with GNNs. The second pipeline directly employs the LLMs to generate predictions.
2. For *LLMs-as-Enhancers*, we introduce two strategies to enhance text attributes via LLMs. We further conduct a series of experiments to compare the effectiveness of these enhancements.
3. For *LLMs-as-Predictors*, we design a series of experiments to explore LLMs' capability in utilizing structural and attribute information. From empirical results, we summarize some original observations and provide new insights.

Key Insights. Through comprehensive empirical evaluations, we find the following key insights:

1. For *LLMs-as-Enhancers*, using deep sentence embedding models to generate embeddings for node attributes show both effectiveness and efficiency.
2. For *LLMs-as-Enhancers*, utilizing LLMs to augment node attributes at the text level also leads to improvements in downstream performance.
3. For *LLMs-as-Predictors*, LLMs present preliminary effectiveness but we should be careful about their inaccurate predictions and the potential test data leakage problem.
4. LLMs demonstrate the potential to serve as good annotators for labeling nodes, as a decent portion of their annotations is accurate.

Organization. The remaining of this paper is organized as follows. Section 2 introduces necessary preliminary knowledge and notations used in this paper. Section 3 introduces two pipelines to leverage LLMs under the task of node classification. Section 4 explores the first pipeline, *LLMs-as-Enhancers*, which adopts LLMs to enhance text attributes. Section 5 details the second pipeline, *LLMs-as-Predictors*, exploring the potential for directly applying LLMs to solve graph learning problems as a predictor. Section 6 discusses works relevant to the applications of LLMs in the graph domain. Section 7 summarizes our insights and discusses the limitations of our study and the potential directions of LLMs in the graph domain.

2. PRELIMINARIES

In this section, we present concepts, notations and problem settings used in the work. We primarily delve into the node classification task on the text-attributed graphs, which is one of the most important downstream tasks in the graph learning domain. Next, we first give the definition of text-attributed graphs.

Text-Attributed Graphs A text-attributed graph (TAG) $\mathcal{G}_S$ is defined as a structure consisting of nodes $\mathcal{V}$ and their corresponding adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$. For each node $v_i \in \mathcal{V}$, it is associated with a text attribute, denoted as $\mathbf{s}_i$.

In this study, we focus on node classification, which is one of the most commonly adopted graph-related tasks.

Node Classification on TAGs Given a set of labeled nodes $\mathcal{L} \subset \mathcal{V}$ with their labels $y_{\mathcal{L}}$, we aim to predict the labels $\mathbf{y}_{\mathcal{U}}$ for the remaining unlabeled nodes $\mathcal{U} = \mathcal{V} \setminus \mathcal{L}$.

We use the citation network dataset Ogbn-arxiv [23] as an illustrative example. In such a graph, each node represents an individual paper from the computer science subcategory, with the attribute of the node embodying the paper's title and abstracts. The edges denote the citation relationships. The task is to classify the papers into their corresponding categories, for example, "cs.cv" (i.e., computer vision). Next, we introduce the models adopted in this study, including graph neural networks and large language models.

Graph Neural Networks. When applied to TAGs for node classification, Graph Neural Networks (GNNs) leverage the structural interactions between nodes. Given initial node features h_i^0, GNNs update the representation of each node by aggregating the information from neighboring nodes in a message-passing manner [16]. The l-th layer can be formulated as:

$$h_i^l = \mathrm{UPD}^l\left(h_i^{l-1}, \mathrm{AGG}_{j\in\mathcal{N}(i)}\,\mathrm{MSG}^l\left(h_i^{l-1}, h_j^{l-1}\right)\right), \quad (1)$$

where AGG is often an aggregation function such as summation, or maximum. UPD and MSG are usually some differentiable functions, such as MLP. The final hidden representations can be passed through a fully connected layer to make classification predictions.

Large Language Models. In this work, we primarily utilize the term "large language models" (LLMs) to denote language models that have been pre-trained on extensive text corpora. Despite the diversity of pre-training objectives [9; 52; 53], the shared goal of these LLMs is to harness the knowledge acquired during the pre-training phase and repurpose it for a range of downstream tasks. Based on their interfaces, specifically considering whether their embeddings are accessible to users or not, in this work we roughly classify LLMs as below:

Embedding-visible LLMs Embedding-visible LLMs pro-

vide access to their embeddings, allowing users to interact with and manipulate the underlying language representations. Embedding-visible LLMs enable users to extract embeddings for specific words, sentences, or documents, and perform various natural language processing tasks using those embeddings. Examples of embedding-visible LLMs include BERT [9], Sentence-BERT [54], and Deberta [21].

Embedding-invisible LLMs Embedding-invisible LLMs do not provide direct access to their embeddings or allow users to manipulate the underlying language representations. Instead, they are typically deployed as web services [59] and offer restricted interfaces. For instance, ChatGPT [45], along with its API, solely provides a text-based interface. Users can only engage with these LLMs through text interactions.

In addition to the interfaces, the size, capability, and model structure are crucial factors in determining how LLMs can be leveraged for graphs. Consequently, we take into account the following four types of LLMs:

1. ***Pre-trained Language Models:*** We use the term "pre-trained language models" (PLMs) to refer to those relatively small large language models, such as Bert [9] and Deberta [21], which can be fine-tuned for downstream tasks. It should be noted that strictly speaking, all LLMs can be viewed as PLMs. Here we adopt the commonly used terminology for models like BERT [51] to distinguish them from other LLMs follwing the convention in a recent paper [85].
2. ***Deep Sentence Embedding Models:*** These models typically use PLMs as the base encoders and adopt the bi-encoder structure [54; 68; 44]. They further pre-train the models in a supervised [54] or contrastive manner [68; 44]. In most cases, there is no need for these models to conduct additional fine-tuning for downstream tasks. These models can be further categorized into *local sentence embedding models* and *online sentence embedding models*. *Local sentence embedding models* are open-source and can be accessed locally, with Sentence-BERT (SBERT) being an example. On the other hand, *online sentence embedding models* are closed-source and deployed as services, with OpenAI's text-ada-embedding-002 [44] being an example.
3. ***Large Language Models:*** Compared to PLMs, Large Language Models (LLMs) exhibit significantly enhanced capabilities with orders of magnitude more parameters. LLMs can be categorized into two types. The first type consists of open-source LLMs, which can be deployed locally, providing users with transparent access to the models' parameters and embeddings. However, the substantial size of these models poses a challenge, as fine-tuning them can be quite cumbersome. One representative example of an open-source LLM is LLaMA [63]. The second type of LLMs is typically deployed as services [59], with restrictions placed on user interfaces. In this case, users are unable to access the model parameters, embeddings, or logits directly. The most powerful LLMs such as ChatGPT [45] and GPT4 [46] belong to this kind.

Among the four types of LLMs, PLMs, deep sentence embedding models, and open-source LLMs are often embedding-visible LLMs. Closed-source LLMs are embedding-invisible LLMs.

3. PIPELINES FOR LLMS IN GRAPHS

Given the superior power of LLMs in understanding textual information, we now investigate different strategies to leverage LLMs for node classification in textual graphs. Specifically, we present two distinct pipelines: *LLMs-as-Enhancers* and *LLMs-as-Predictors*. Figure 1 provides figurative illustrations of these two pipelines, and we elaborate on their details as follows.

LLMs-as-Enhancers In this pipeline, LLMs are leveraged to enhance the text attributes. As shown in Figure 1, for *LLMs-as-Enhancers*, LLMs are adopted to pre-process the text attributes, and then GNNs are trained on the enhanced attributes as the predictors. Considering different structures of LLMs, we conduct enhancements either at the **feature level** or at the **text level** as shown in Figure 2.

1. ***Feature-level enhancement:*** For feature-level enhancement, embedding-visible LLMs inject their knowledge by simply encoding the text attribute s_i into text embeddings $h_i \in R^d$. We investigate two feasible **integrating structures** for feature-level enhancement. **(1) Cascading structure:** Embedding-visible LLMs and GNNs are combined sequentially. Embedding-visible LLMs first encode text attributes into text features, which are then adopted as the initial node features for GNNs. **(2) Iterative structure [83]:** PLMs and GNNs are co-trained together by generating pseudo labels for each other. Only PLMs are suitable for this structure since it involves fine-tuning.
2. ***Text-level enhancement:*** For text-level enhancement, given the text attribute s_i, LLMs will first transform the text attribute into augmented attribute s_i^{Aug}. Enhanced attributes will then be encoded into enhanced node features $h_i^{Aug} \in R^d$ through embedding-visible LLMs. GNNs will make predictions by ensembling the original node features and augmented node features.

LLMs-as-Predictors In this pipeline, LLMs are leveraged to directly make predictions for the node classification task. As shown in Figure 1b, for *LLMs-as-Predictors*, the first step is to design prompts to represent graph structural information, text attributes, and label information with texts. Then, embedding-invisible LLMs make predictions based on the information embedded in the prompts.

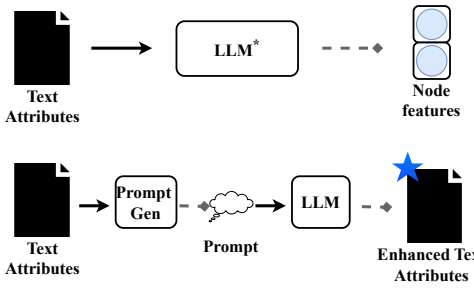
4. LLMS AS THE ENHANCERS

In this section, we investigate the potential of employing LLMs to enrich the text attributes of nodes. As presented in Section 3, we consider ***feature-level enhancement***, which injects LLMs' knowledge by encoding text attributes into features. Moreover, we consider ***text-level enhancement***, which inject LLMs' knowledge by augmenting the text attributes at the text level. We first study ***feature-level enhancement***.

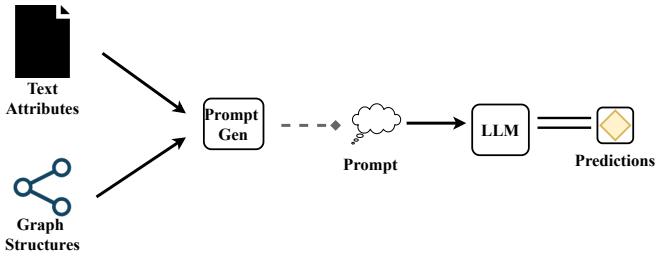
4.1 Feature-level Enhancement

In *feature-level enhancement*, we mainly study how to combine embedding-visible LLMs with GNNs at the feature level. The embedding generated by LLMs will be adopted as the initial features of GNNs. We first briefly introduce the dataset and dataset split settings we use.

Datasets. In this study, we adopt CORA [40], PUBMED [57], OGBN-ARXIV, and OGBN-PRODUCTS [23], four popular benchmarks for node classification. We present their detailed statistics and descriptions in Appendix A. Specifically, we



(a) *An illustration of LLMs-as-Enhancers*, where LLMs preprocess the text attributes, and GNNs eventually make the predictions. Three different structures for this pipeline are demonstrated in Figure 2.



(b) *An illustration of LLMs-as-Predictors*, where LLMs directly make the predictions. The key component for this pipeline is how to design an effective prompt to incorporate structural and attribute information.

Figure 1: Pipelines for integrating LLMs into graph learning. In all figures, we use "PLM" to denote small-scale PLMs that can be fine-tuned on downstream datasets, "LLM*" to denote embedding-visible LLMs, and "LLM" to denote embedding-invisible LLMs.

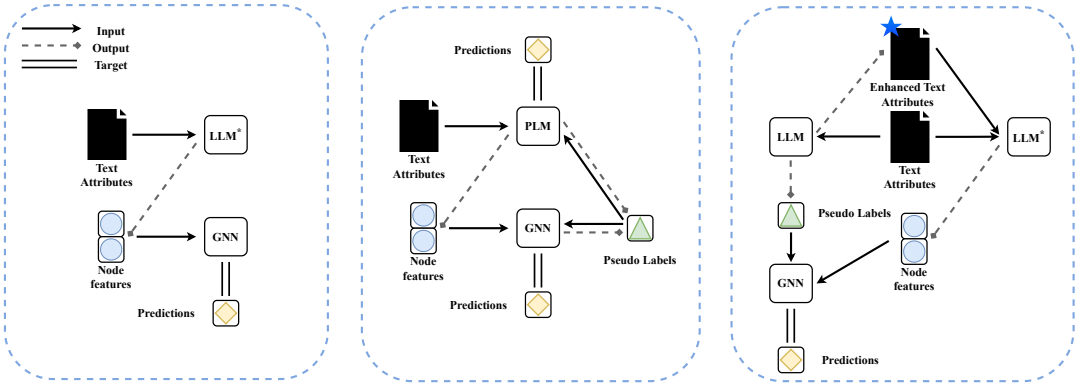


Figure 2: Three strategies to adopt LLMs as enhancers. The first two integrating structures are designed for feature-level enhancement, while the last structure is designed for text-level enhancement. From left to right: (1) Cascading Structure: Embedding-visible LLMs enhance text attributes directly by encoding them into initial node features for GNNs. (2) Iterative Structure: GNNs and PLMs are co-trained in an iterative manner. (3) Text-level enhancement structure: Embedding-invisible LLMs are initially adopted to enhance the text attributes by generating augmented attributes. The augmented attributes and original attributes are encoded and then ensembled together.

examine two classification dataset split settings, specifically tailored for the CORA and PUBMED datasets. Meanwhile, for OGBN-ARXIV and OGBN-PRODUCTS, we adopt the official dataset splits. (1) For CORA and PUBMED, the first splitting setting addresses **low-labeling-rate** conditions, which is a commonly adopted setting [75]. To elaborate, we randomly select 20 nodes from each class to form the training set. Then, 500 nodes are chosen for the validation set, while 1000 additional random nodes from the remaining pool are used for the test set. (2) The second splitting setting caters to **high-labeling-rate** scenarios, which is also a commonly used setting, and also adopted by TAPE [22]. In this setting, 60% of the nodes are designated for the training set, 20% for the validation set, and the remaining 20% are set aside for the test set. We take the output of GNNs and compare it with the ground truth of the dataset. We conduct all the experiments on 10 different seeds and report both average accuracy and variance.

Baseline Models. In our exploration of how LLMs augment node attributes at the feature level, we consider three main components: (1) *Selection of GNNs*, (2) *Selection of LLMs*, and (3) *Intergrating structures for LLMs and GNNs*. In this study, we choose the most representative models for each component, and the details are listed below.

1. *Selection of GNNs:* For GNNs on CORA and PUBMED, we consider Graph Convolutional Network (GCN) [27] and Graph Attention Network (GAT) [64]. We also include the performance of MLP to **evaluate the quality of text embeddings without aggregations**. For OGBN-ARXIV, we consider GCN, MLP, and a better-performed GNN model RevGAT [28]. For OGBN-PRODUCTS, we consider GraphSAGE [19] which supports neighborhood sampling for large graphs, MLP, and a state-of-the-art model SAGN [58]. For RevGAT and SAGN, we adopt all tricks utilized in the OGB leaderboard [23][1].
2. *Selection of LLMs:* To enhance the text attributes at the feature level, we specifically require embedding-visible LLMs. Specifically, we select (1) **Fixed PLM/LLMs without fine-tuning:** We consider Deberta [21] and LLaMA [63]. The first one is adapted from GLEM [83] and we follow the setting of GLEM [83] to adopt the

[1] https://ogb.stanford.edu/docs/leader_nodeprop/

[CLS] token of PLMs as the text embeddings. LLaMA is a widely adopted open-source LLM, which has also been included in Langchain[2]. We adopt LLaMA-cpp[3], which adopt the [EOS] token as text embeddings in our experiments. (2) **Local sentence embedding models**: We adopt Sentence-BERT [54] and e5-large [68]. The former is one of the most popular lightweight deep text embedding models while the latter is the state-of-the-art model on the MTEB leaderboard [43]. (3) **Online sentence embedding models**: We consider two online sentence embedding models, i.e., text-ada-embedding-002 [44] from OpenAI, and Palm-Cortex-001 [1] from Google. Although the strategy to train these models has been discussed [1; 44], their detailed parameters are not known to the public, together with their capability on node classification tasks. (4) **Fine-tuned PLMs**: We consider fine-tuning Deberta on the downstream dataset, and also adopt the last hidden states of PLMs as the text embeddings. For fine-tuning, we consider two integrating structures below.

3. *Integration structures:* We consider **cascading structure** and **iterative structure**. (1) **Cascading structure:** we first fine-tune the PLMs on the downstream dataset. Subsequently, the text embeddings engendered by the fine-tuned PLM are employed as the initial node features for GNNs. (2) **Iterative structure:** PLMs and GNNs are first trained separately and further co-trained in an iterative manner by generating pseudo labels for each other. This grants us the flexibility to choose either the final iteration of PLMs or GNNs as the predictive models, which are denoted as "GLEM-LM" and "GLEM-GNN", respectively.

We also consider non-contextualized shallow embeddings [41] including TF-IDF and Word2vec [23] as a comparison. TF-IDF is adopted to process the original text attributes for PUBMED [57], and Word2vec is utilized to encode the original text attributes for OGBN-ARXIV [23]. For OGBN-ARXIV and OGBN-PRODUCTS, we also consider the GIANT features [6], which can not be directly applied to CORA and PUBMED because of its special pre-training strategy. Furthermore, we don't include LLaMA for OGBN-ARXIV and OGBN-PRODUCTS because it imposes an excessive computational burden when dealing with large-scale datasets.

The results are shown in Table 1, Table 2, and Table 3. In these tables, we demonstrate the performance of different combinations of text encoders and GNNs. We also include the performance of MLPs which can suggest the original quality of the textual embeddings before the aggregation. Moreover, We use colors to show the top 3 best LLMs under each GNN (or MLP) model. Specifically, We use yellow to denote the best one under a specific GNN/MLP model, green the second best one, and pink the third best one.

4.1.1 Node Classification Performance Comparison

Observation 1. Combined with different types of text embeddings, GNNs demonstrate distinct effectiveness.

From Table 3, if we compare the performance of TF-IDF and fine-tuned PLM embeddings when MLP is the predictor, we can see that the latter usually achieves much better performance. However, when a GNN model is adopted as the predictor, the performance of TF-IDF embedding is close to and even surpasses the PLM embedding. This result is consistent with the findings in [49], which suggests that GNNs present distinct effectiveness for different types of text embeddings. However, we don't find a simple metric to determine the effectiveness of GNNs on different text embeddings. We will further discuss this limitation in Section 7.2.

Observation 2. Fine-tune-based LLMs may fail at low labeling rate settings.

From Table 1, we note that no matter the cascading structure or the iterative structure, fine-tune-based LLMs' embeddings perform poorly for low labeling rate settings. Both fine-tuned PLM and GLEM present a large gap against deep sentence embedding models and TF-IDF, which do not involve fine-tuning. When training samples are limited, fine-tuning may fail to transfer sufficient knowledge for the downstream tasks.

Observation 3. With a simple cascading structure, the combination of deep sentence embedding with GNNs makes a strong baseline.

From Table 1, Table 2, Table 3, we can see that with a simple cascading structure, the combination of deep sentence embedding models (including both local sentence embedding models and online sentence embedding models) with GNNs show competitive performance, under all dataset split settings. The intriguing aspect is that, during the pre-training stage of these deep sentence embedding models, no structural information is incorporated. Therefore, it is astonishing that these structure-unaware models can outperform GIANT on OGBN-ARXIV, which entails a structure-aware self-supervised learning stage.

Observation 4. Simply enlarging the model size of LLMs may not help with the node classification performance.

From Table 1 and Table 2, we can see that although the performance of the embeddings generated by LLaMA outperforms the Deberta-base without fine-tuning by a large margin, there is still a large performance gap between the performance of embeddings generated by deep sentence embedding models in the low labeling rate setting. This result indicates that simply increasing the model size may not be sufficient to generate high-quality embeddings for node classification. The pre-training objective may be an important factor.

4.1.2 Scalability Investigation

In the aforementioned experimental process, we empirically find that in larger datasets like OGBN-ARXIV, methods like GLEM that require fine-tuning of the PLMs will take several orders of magnitude more time in the training stage than these that do not require fine-tuning. It presents a hurdle for these approaches to be applied to even larger datasets or scenarios with limited computing resources. To gain a more comprehensive understanding of the efficiency and scalability of different LLMs and integrating structures, we conduct an experiment to measure the running time and memory usage of different approaches. It should be noted that we mainly consider the scalability problem in the training stage, which is different from the efficiency problem in the inference stage.

In this study, we choose representative models from each type of LLMs, and each kind of integrating structure. For

[2] `https://python.langchain.com/`

[3] `https://github.com/ggerganov/llama.cpp`

Table 1: Experimental results for feature-level *LLMs-as-Enhancer* on CORA and PUBMED with a low labeling ratio. Since MLPs do not provide structural information, it is meaningless to co-train it with PLM (with their performance shown as N/A). We use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

	CORA			PUBMED		
	GCN	GAT	MLP	GCN	GAT	MLP
Non-contextualized Shallow Embeddings						
TF-IDF	81.99 ± 0.63	82.30 ± 0.65	67.18 ± 1.01	78.86 ± 2.00	77.65 ± 0.91	71.07 ± 0.78
Word2Vec	74.01 ± 1.24	72.32 ± 0.17	55.34 ± 1.31	70.10 ± 1.80	69.30 ± 0.66	63.48 ± 0.54
PLM/LLM Embeddings without Fine-tuning						
Deberta-base	48.49 ± 1.86	51.02 ± 1.22	30.40 ± 0.57	62.08 ± 0.06	62.63 ± 0.27	53.50 ± 0.43
LLama 7B	66.80 ± 2.20	59.74 ± 1.53	52.88 ± 1.96	73.53 ± 0.06	67.52 ± 0.07	66.07 ± 0.56
Local Sentence Embedding Models						
Sentence-BERT(MiniLM)	82.20 ± 0.49	82.77 ± 0.59	74.26 ± 1.44	81.01 ± 1.32	79.08 ± 0.07	76.66 ± 0.50
e5-large	82.56 ± 0.73	81.62 ± 1.09	74.26 ± 0.93	82.63 ± 1.13	79.67 ± 0.80	80.38 ± 1.94
Online Sentence Embedding Models						
text-ada-embedding-002	82.72 ± 0.69	82.51 ± 0.86	73.15 ± 0.89	79.09 ± 1.51	80.27 ± 0.41	78.03 ± 1.02
Google Palm Cortex 001	81.15 ± 1.01	82.79 ± 0.41	69.51 ± 0.83	80.91 ± 0.19	80.72 ± 0.33	78.93 ± 0.90
Fine-tuned PLM Embeddings						
Fine-tuned Deberta-base	59.23 ± 1.16	57.38 ± 2.01	30.98 ± 0.68	62.12 ± 0.07	61.57 ± 0.07	53.65 ± 0.26
Iterative Structure						
GLEM-GNN	48.49 ± 1.86	51.02 ± 1.22	N/A	62.08 ± 0.06	62.63 ± 0.27	N/A
GLEM-LM	59.23 ± 1.16	57.38 ± 2.01	N/A	62.12 ± 0.07	61.57 ± 0.07	N/A

Table 2: Experimental results for feature-level *LLMs-as-Enhancers* on CORA and PUBMED with a high labeling ratio. We use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

	CORA			PUBMED		
	GCN	GAT	MLP	GCN	GAT	MLP
Non-contextualized Shallow Embeddings						
TF-IDF	90.90 ± 2.74	90.64 ± 3.08	83.98 ± 5.91	89.16 ± 1.25	89.00 ± 1.67	89.72 ± 3.57
Word2Vec	88.40 ± 2.25	87.62 ± 3.83	78.71 ± 6.32	85.50 ± 0.77	85.63 ± 0.93	83.80 ± 1.33
PLM/LLM Embeddings without Fine-tuning						
Deberta-base	65.86 ± 1.96	79.67 ± 3.19	45.64 ± 4.41	67.33 ± 0.69	67.81 ± 1.05	65.07 ± 0.57
LLama 7B	89.69 ± 1.86	87.66 ± 4.84	80.66 ± 7.72	88.26 ± 0.78	88.31 ± 2.01	89.39 ± 1.09
Local Sentence Embedding Models						
Sentence-BERT(MiniLM)	89.61 ± 3.23	90.68 ± 2.22	86.45 ± 5.56	90.32 ± 0.91	90.80 ± 2.02	90.59 ± 1.23
e5-large	90.53 ± 2.33	89.10 ± 3.22	86.19 ± 4.38	89.65 ± 0.85	89.55 ± 1.16	91.39 ± 0.47
Online Sentence Embedding Models						
text-ada-embedding-002	89.13 ± 2.00	90.42 ± 2.50	85.97 ± 5.58	89.81 ± 0.85	91.48 ± 1.94	92.63 ± 1.14
Google Palm Cortex 001	90.02 ± 1.86	90.31 ± 2.82	81.03 ± 2.60	89.78 ± 0.95	90.52 ± 1.35	91.87 ± 0.84
Fine-tuned PLM Embeddings						
Fine-tuned Deberta-base	85.86 ± 2.28	86.52 ± 1.87	78.20 ± 2.25	91.49 ± 1.92	89.88 ± 4.63	94.65 ± 0.13
Iterative Structure						
GLEM-GNN	89.13 ± 0.73	88.95 ± 0.64	N/A	92.57 ± 0.25	92.78 ± 0.21	N/A
GLEM-LM	82.71 ± 1.08	83.54 ± 0.99	N/A	94.36 ± 0.21	94.62 ± 0.14	N/A

TF-IDF, it's a shallow embedding that doesn't involve either training or inference, so the time and memory complexity of the LM phase can be neglected. In terms of Sentence-BERT, for the LM phase, this kind of local sentence embedding model does not involve a fine-tuning stage, and they only need to generate the initial embeddings. For text-ada-embedding-002, which is offered as an API service, we make API calls to generate embeddings. In this part, we set the batch size of Ada to 1,024 and call the API asynchronously, then we measure the time consumption to generate embeddings as the LM phase running time. For Deberta-base, we record the time used to fine-tune the model and generate the text embeddings as the LM phase running time. For GLEM, since it co-trains the PLM and GNNs, we consider LM phase running time and GNN phase running time together (and show the total training time in the "LM phase" column). The efficiency results are shown in Table 4. We also report the peak memory usage in the table. We adopt the default output dimension of each text encoder, which is shown in the brackets.

Observation 5. For integrating structures, iterative structure introduces massive computation overhead in the training stage.

From Table 2 and Table 3, GLEM presents a superior performance in datasets with an adequate number of labeled training samples, especially in large-scale datasets like OGBN-ARXIV and OGBN-PRODUCTS. However, from Table 4, we can see that it introduces massive computation overhead in the training stage compared to Deberta-base with a cascading structure, which indicates the potential efficiency problem of the iterative structures.

Moreover, from Table 4, we note that for the GNN phase,

Table 3: Experimental results for feature-level *LLMs-as-Enhancers* on OGBN-ARXIV and OGBN-PRODUCTS dataset. MLPs do not provide structural information so it's meaningless to co-train it with PLM, thus we don't show the performance. We use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

	OGBN-ARXIV			OGBN-PRODUCTS		
	GCN	MLP	RevGAT	SAGE	SAGN	MLP
Non-contextualized Shallow Embeddings						
TF-IDF	72.23 ± 0.21	66.60 ± 0.25	75.16 ± 0.14	79.73 ± 0.48	84.40 ± 0.07	64.42 ± 0.18
Word2Vec	71.74 ± 0.29	55.50 ± 0.23	73.78 ± 0.19	81.33 ± 0.79	84.12 ± 0.18	69.27 ± 0.54
PLM/LLM Embeddings without Fine-tuning						
Deberta-base	45.70 ± 5.59	40.33 ± 4.53	71.20 ± 0.48	62.03 ± 8.82	74.90 ± 0.48	7.18 ± 1.09
Local Sentence Embedding Models						
Sentence-BERT(MiniLM)	73.10 ± 0.25	71.62 ± 0.10	76.94 ± 0.11	82.51 ± 0.53	84.79 ± 0.23	72.73 ± 0.34
e5-large	73.74 ± 0.12	72.75 ± 0.00	76.59 ± 0.44	82.46 ± 0.91	85.47 ± 0.21	77.49 ± 0.29
Online Sentence Embedding Models						
text-ada-embedding-002	72.76 ± 0.23	72.17 ± 0.00	76.64 ± 0.20	82.90 ± 0.42	85.20 ± 0.19	76.42 ± 0.31
Fine-tuned PLM Embeddings						
Fine-tuned Deberta-base	74.65 ± 0.12	72.90 ± 0.11	75.80 ± 0.39	82.15 ± 0.16	84.01 ± 0.05	79.08 ± 0.23
Others						
GIANT	73.29 ± 0.10	73.06 ± 0.11	75.90 ± 0.19	83.16 ± 0.19	86.67 ± 0.09	79.82 ± 0.07
Iterative Structure						
GLEM-GNN	75.93 ± 0.19	N/A	76.97 ± 0.19	83.16 ± 0.09	87.36 ± 0.07	N/A
GLEM-LM	75.71 ± 0.24	N/A	75.45 ± 0.12	81.25 ± 0.15	84.83 ± 0.04	N/A

Table 4: Efficiency analysis on OGBN-ARXIV. Note that we show the dimension of generated embeddings in the brackets. For GIANT, it adopts a special pre-training stage, which will introduce computation overhead with orders of magnitude larger than that of fine-tuning. The specific time was not discussed in the original paper, therefore its cost in LM-phase is not shown in the table.

Input features	Backbone	LM-phase Running time(s)	LM-phase Memory (GB)	GNN-phase Running time(s)	GNN-phase Memory (GB)
TF-IDF (1024)	GCN	N/A	N/A	53	9.81
	RevGAT	N/A	N/A	873	7.32
Sentence-BERT (384)	GCN	239	1.30	48	7.11
	RevGAT	239	1.30	674	4.37
text-ada-embedding-002 (1536)	GCN	165	N/A	73	11.00
	RevGAT	165	N/A	1038	8.33
Deberta-base (768)	GCN	13560	12.53	50	9.60
	RevGAT	13560	12.53	122	6.82
GLEM-GNN (768)	GCN	68071	18.22	N/A	N/A
	RevGAT	68294	18.22	N/A	N/A
GIANT (768)	GCN	N/A	N/A	50	9.60
	RevGAT	N/A	N/A	122	6.82

the dimension of initial node features, which is the default output dimension of text encoders mainly determines memory usage and time cost.

Observation 6. In terms of different LLM types, deep sentence embedding models present better efficiency in the training stage.

In Table 4, we analyze the efficiency of different types of LLMs by selecting representative models from each category. Comparing fine-tune-based PLMs with deep sentence embedding models, we observe that the latter demonstrates significantly better time efficiency as they do not require a fine-tuning stage. Additionally, deep sentence embedding models exhibit improved memory efficiency as they solely involve the inference stage without the need to store additional information such as gradients.

4.2 Text-level Enhancement

For feature-level enhancement, LLMs in the pipeline must be embedding-visible. However, the most powerful LLMs such as ChatGPT [45], PaLM [1], and GPT4 [46] are all deployed as online services [59], which put strict restrictions so that users can not get access to model parameters and embeddings. Users can only interact with these embedding-invisible LLMs through texts, which means that user inputs must be formatted as texts and LLMs will only yield text outputs. In this section, we explore the potential for these embedding-invisible LLMs to do text-level enhancement. To enhance the text attribute at the text level, the key is to expand more information that is not contained in the original text attributes. Based on this motivation and a recent paper [22], we study the following two potential text-level enhancements, and illustrative examples of these two augmentations are shown in Figure 3.

1. **TAPE** [22]: The motivation of TAPE is to leverage the knowledge of LLMs to generate high-quality node features. Specifically, it uses LLMs to generate pseudo labels

and explanations. These explanations aim to make the logical relationship between the text features and corresponding labels more clear. For example, given the original attributes "mean-field approximation" and the ground truth label "probabilistic methods", it will generate a description such as "mean-field approximation is a widely adopted simplification technique for probabilistic models", which makes the connection of these two attributes much more clear. After generating pseudo labels and explanations, they further adopt PLMs to be fine-tuned on both the original text attributes and the explanations generated by LLMs, separately. Next, they generate the corresponding text features and augmented text features based on the original text attributes and augmented text attributes respectively, and finally ensemble them together as the initial node features for GNNs.

2. **Knowledge-Enhanced Augmentation**: The motivation behind Knowledge-Enhanced Augmentation (KEA) is to enrich the text attributes by providing additional information. KEA is inspired by knowledge-enhanced PLMs such as ERNIE [61] and K-BERT [36] and aims to explicitly incorporate external knowledge. In KEA, we prompt the LLMs to generate a list of knowledge entities along with their text descriptions. For example, we can generate a description for the abstract term "Hopf-Rinow theorem" as follows: "The Hopf-Rinow theorem establishes that a Riemannian manifold, which is both complete and connected, is geodesically complete if and only if it is simply connected." By providing such descriptions, we establish a clearer connection between the theorem and the category "Riemannian geometry". Once we obtain the entity list, we encode it either together with the original text attribute or separately. We try encoding text attributes with fine-tuned PLMs and deep sentence embedding models. We also employ ensemble methods to combine these embeddings. One potential advantage of KEA is that it is loosely coupled with the prediction performance of LLMs. In cases where LLMs generate incorrect predictions, TAPE may potentially generate low-quality node features because the explanations provided by PLMs may also be incorrect. However, with KEA, the augmented features may exhibit better stability since we do not rely on explicit predictions from LLMs.

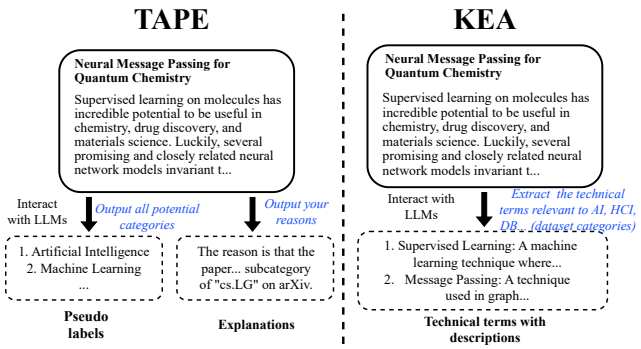


Figure 3: Illustrations for TAPE and KEA. TAPE leverages the knowledge of LLMs to generate explanations for their predictions. For KEA, we prompt the LLMs to generate a list of technical terms with their descriptions. The main motivation is to augment the attribute information.

4.2.1 *Experimental Setups*

To evaluate these two strategies, we conduct experiments on two small datasets CORA and PUBMED considering the cost to use the LLMs. For low labeling ratio and high labeling ratio, we adopt the same setting as that in Table 1 and Table 2. For predictors, we adopt GCN, GAT, and MLP to study both the quality of textual embeddings before and after aggregations. For LLMs, we adopt ChatGPT with the latest version (gpt-3.5-turbo-0613). To better understand the effectiveness of TAPE, we separate it into TA, P, and E, where "TA" refers to "text attributes", "P" refers to "pseudo labels", and "E" refers to "explanations". For KEA, we try two approaches to inject the augmented textual attributes. The first approach is appending the augmented textual attributes into the original attribute, which is denoted as "KEA-I". Then the combined attributes are encoded into features. The second approach is to encode the augmented attributes and original attributes separately, which is denoted as "KEA-S". We report the results for original, augmented, and ensembling features. Both TAPE and KEA adopt the cascading structures. After encoding the text attributes with LLMs, the generated embeddings are adopted as the initial features for GNNs. We try two approaches to encode the attributes, which are fine-tuned PLMs and local sentence embedding models. Specifically, we adopt Deberta-base and e5-large. To conduct a fair comparison, we first determine the better text encoder by evaluating their overall performance. Once the text encoder is selected, we proceed to compare the performance of the augmented attributes against the original attributes.

A comprehensive evaluation of TAPE. We first gain a deeper understanding of TAPE through a comprehensive ablation study. The experimental results are shown in Table 5 and Table 6. We show the approach we adopt to encode the text attributes in the bracket. In particular, we mainly consider fine-tuned Deberta-base, which is denoted as PLM, and e5-large, which is denoted as e5.

Observation 7. The effectiveness of TAPE is mainly from the explanations E generated by LLMs.

From the ablation study, we can see that compared to pseudo labels **P**, the explanations present better stability across different datasets. One main advantage of adopting explanations generated by LLMs is that these augmented attributes present better performance in the low-labeling rate setting. From Table 5, we note that when choosing PLM as the encoders, **E** performs much better than **TA** in the low labeling rate setting. Compared to explanations, we find that the effectiveness of the P mainly depends on the zero-shot performance of LLMs, which may present large variances across different datasets. In the following analysis, we use **TA** + **E** and neglect the pseudo labels generated by LLMs.

Observation 8. Replacing fine-tuned PLMs with deep sentence embedding models can further improve the overall performance of TAPE.

From Table 5 and Table 6, we observe that adopting e5-large as the LLMs to encode the text attributes can achieve good performance across different datasets and different data splits. Specifically, the **TA** + **E** encoded with e5 can achieve top 3 performance in almost all settings. In the following analysis, we adopt e5 to encode the original and enhanced attributes **TA** + **E**.

Table 5: A detailed ablation study of TAPE on CORA and PUBMED dataset in low labeling rate setting. For each combination of features and models, we use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

		CORA			PUBMED		
		GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
TAPE	**TAPE**	74.56 ± 2.03	75.27 ± 2.10	64.44 ± 0.60	85.97 ± 0.31	86.97 ± 0.33	93.18 ± 0.28
	P	52.79 ± 1.47	62.13 ± 1.50	63.56 ± 0.52	81.92 ± 1.89	88.27 ± 0.01	93.27 ± 0.15
	TA + E (e5)	83.38 ± 0.42	84.00 ± 0.09	75.73 ± 0.53	87.44 ± 0.49	86.71 ± 0.92	90.25 ± 1.56
	TA + E (PLM)	78.02 ± 0.56	64.08 ± 12.36	55.72 ± 11.98	80.70 ± 1.73	79.66 ± 3.08	76.42 ± 2.18
	E (PLM)	79.46 ± 1.10	74.82 ± 1.19	63.04 ± 0.88	81.88 ± 0.05	81.56 ± 0.07	76.90 ± 1.60
	E (e5)	84.38 ± 0.36	83.01 ± 0.60	70.64 ± 1.10	82.23 ± 0.78	80.30 ± 0.77	77.23 ± 0.48
Original attributes	**TA** (PLM)	59.23 ± 1.16	57.38 ± 2.01	30.98 ± 0.68	62.12 ± 0.07	61.57 ± 0.07	53.65 ± 0.26
	TA (e5)	82.56 ± 0.73	81.62 ± 1.09	74.26 ± 0.93	82.63 ± 1.13	79.67 ± 0.80	80.38 ± 1.94

Table 6: A detailed ablation study of TAPE on CORA and PUBMED dataset in the high labeling rate setting. For each combination of features and models, we use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

		CORA			PUBMED		
		GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
TAPE	**TAPE**	87.88 ± 0.98	88.69 ± 1.13	83.09 ± 0.91	92.22 ± 1.30	93.35 ± 1.50	95.05 ± 0.27
	P	64.90 ± 1.39	80.11 ± 4.01	70.31 ± 1.91	85.73 ± 0.59	91.60 ± 0.62	93.65 ± 0.35
	TA + E (e5)	90.68 ± 2.12	91.86 ± 1.36	87.00 ± 4.83	92.64 ± 1.00	93.35 ± 1.24	94.34 ± 0.86
	TA + E (PLM)	87.44 ± 1.74	88.40 ± 1.60	82.80 ± 1.00	90.23 ± 0.71	91.73 ± 1.58	95.40 ± 0.32
	E (PLM)	83.28 ± 4.53	82.47 ± 6.06	80.41 ± 3.35	88.90 ± 2.94	83.00 ± 14.07	87.75 ± 14.83
	E (e5)	89.39 ± 2.69	90.13 ± 2.52	84.05 ± 4.03	89.68 ± 0.78	90.61 ± 1.61	91.09 ± 0.85
Original attributes	**TA** (PLM)	85.86 ± 2.28	86.52 ± 1.87	78.20 ± 2.25	91.49 ± 1.92	89.88 ± 4.63	94.65 ± 0.13
	TA (e5)	90.53 ± 2.33	89.10 ± 3.22	86.19 ± 4.38	89.65 ± 0.85	89.55 ± 1.16	91.39 ± 0.47

4.2.1.1 *Effectiveness of* KEA.

We then show the results of **KEA** in Table 7 and Table 8. For **KEA-I**, we inject the description of each technical term directly into the original attribute. For **KEA-S**, we encode the generated description and original attribute separately.

Observation 9. The proposed knowledge enhancement attributes KEA can enhance the performance of the original attribute TA.

From Table 7 and Table 8, we first compare the performance of features encoded by e5 and PLM. We see that the proposed **KEA** is more fitted to the e5 encoder, and fine-tuned PLM embeddings present poor performance on the low labeling rate, thus we also select e5 as the encoder to further compare the quality of attributes. From Table 9 we can see that the proposed **KEA-I + TA** and **KEA-S + TA** attributes can consistently outperform the original attributes **TA**.

Observation 10. For different datasets, the most effective enhancement methods may vary.

Moreover, we compare the performance of our proposed **KEA** with **TA + E**, and the results are shown in Table 9. We can see that on CORA, our methods can achieve better performance while **TA + E** can achieve better performance on PUBMED. One potential explanation for this phenomenon is that **TA + E** relies more on the capability of LLMs. Although we have removed the pseudo labels **P**, we find that the explanations still contain LLMs' predictions. As a result, the effectiveness of **TA + E** will be influenced by LLMs' performance on the dataset. As shown in [22], the LLMs can achieve superior performance on the PUBMED dataset but perform poorly on the CORA dataset. Compared to **TA + E**, our proposed **KEA** only utilizes the commonsense knowledge of the LLMs, which may have better stability across different datasets.

5. LLMS AS THE PREDICTORS

In the *LLMs-as-Enhancers* pipeline, the role of the LLMs remains somewhat limited since we only utilize their pre-trained knowledge but overlook their reasoning capability. Drawing inspiration from the LLMs' proficiency in handling complex tasks with implicit structures, such as logical reasoning [7] and recommendation [14], we question: **Is it possible for the LLM to independently perform predictive tasks on graph structures?** By shifting our focus to node attributes and overlooking the graph structures, we can perceive node classification as a text classification problem. In [60], the LLMs demonstrate significant promise, suggesting that they can proficiently process text attributes. However, one key problem is that LLMs are not originally designed to process graph structures. Therefore, it can not directly process structural information like GNNs.

In this section, we aim to explore the potential of LLMs as a predictor. In particular, we first check whether LLM can perform well without any structural information. Then, we further explore some prompts to incorporate structural information with natural languages. Finally, we show a case study in Section 5.3 to explore its potential usage as an annotator for graphs.

5.1 How Can LLM Perform on Popular Graph Benchmarks without Structural Information?

In this subsection, we treat the node classification problem as a text classification problem by ignoring the structural information. We adopt ChatGPT (gpt-3.5-turbo-0613) as the LLMs to conduct all the experiments. We choose five popular textual graph datasets with raw text attributes: CORA [40], CITESEER [15], PUBMED [57], OGBN-ARXIV, and OGBN-PRODUCTS [23]. The details of these datasets can be found in Appendix A. Considering the costs to query LLMs'

Table 7: A detailed ablation study of KEA on Cora and Pubmed dataset in the low labeling rate setting. For each combination of features and models, we use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

		Cora			Pubmed		
		GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
Original attributes	TA (PLM)	59.23 ± 1.16	57.38 ± 2.01	30.98 ± 0.68	62.12 ± 0.07	61.57 ± 0.07	53.65 ± 0.26
	TA (e5)	82.56 ± 0.73	81.62 ± 1.09	74.26 ± 0.93	82.63 ± 1.13	79.67 ± 0.80	80.38 ± 1.94
KEA	KEA-I + TA (e5)	83.20 ± 0.56	83.38 ± 0.63	74.34 ± 0.97	83.30 ± 1.75	81.16 ± 0.87	80.74 ± 2.44
	KEA-I + TA (PLM)	53.21 ± 11.54	55.38 ± 4.64	31.80 ± 3.63	57.13 ± 8.20	58.66 ± 4.27	52.28 ± 4.47
	KEA-I (e5)	81.35 ± 0.77	82.04 ± 0.72	70.64 ± 1.10	81.98 ± 0.91	81.04 ± 1.39	79.73 ± 1.63
	KEA-I (PLM)	36.68 ± 18.63	37.69 ± 12.79	30.46 ± 0.60	56.22 ± 7.17	59.33 ± 1.69	52.79 ± 0.51
	KEA-S + TA (e5)	84.63 ± 0.58	85.02 ± 0.40	76.11 ± 2.66	82.93 ± 2.38	81.34 ± 1.51	80.74 ± 2.44
	KEA-S + TA (PLM)	51.36 ± 16.13	52.85 ± 7.00	34.56 ± 5.09	59.47 ± 6.09	51.93 ± 3.27	51.11 ± 2.63
	KEA-S (e5)	84.38 ± 0.36	83.01 ± 0.60	70.64 ± 1.10	82.23 ± 0.78	80.30 ± 0.77	77.23 ± 0.48
	KEA-S (PLM)	28.97 ± 18.24	43.88 ± 10.31	30.36 ± 0.58	61.22 ± 0.94	54.93 ± 1.55	47.94 ± 0.89

Table 8: A detailed ablation study of KEA on Cora and Pubmed dataset in the high labeling rate setting. For each combination of features and models, we use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

		Cora			Pubmed		
		GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
Original Attributes	**TA** (PLM)	85.86 ± 2.28	86.52 ± 1.87	78.20 ± 2.25	91.49 ± 1.92	89.88 ± 4.63	94.65 ± 0.13
	TA (e5)	90.53 ± 2.33	89.10 ± 3.22	86.19 ± 4.38	89.65 ± 0.85	89.55 ± 1.16	91.39 ± 0.47
KEA	**KEA-I + TA** (e5)	91.12 ± 1.76	90.24 ± 2.93	87.88 ± 4.44	90.19 ± 0.83	90.60 ± 1.22	92.12 ± 0.74
	KEA-I + TA (PLM)	87.07 ± 1.04	87.66 ± 0.86	79.12 ± 2.77	92.32 ± 0.64	92.29 ± 1.43	94.85 ± 0.20
	KEA-I (e5)	91.09 ± 1.78	90.13 ± 2.76	86.78 ± 4.12	89.56 ± 0.82	90.25 ± 1.34	91.92 ± 0.80
	KEA-I (PLM)	86.08 ± 2.35	85.23 ± 3.15	77.97 ± 2.87	91.73 ± 0.58	91.93 ± 1.76	94.76 ± 0.33
	KEA-S + TA (e5)	91.09 ± 1.78	92.30 ± 1.69	88.95 ± 4.96	90.40 ± 0.92	90.82 ± 1.30	91.78 ± 0.56
	KEA-S + TA (PLM)	83.98 ± 5.13	87.33 ± 1.68	80.04 ± 1.32	86.11 ± 5.68	89.04 ± 5.82	94.35 ± 0.48
	KEA-S (e5)	89.39 ± 2.69	90.13 ± 2.52	84.05 ± 4.03	89.68 ± 0.78	90.61 ± 1.61	91.09 ± 0.85
	KEA-S (PLM)	83.35 ± 7.30	85.67 ± 2.00	76.76 ± 1.82	79.68 ± 19.57	69.90 ± 19.75	85.91 ± 6.47

Table 9: Comparison of the performance of TA, KEA-I, and KEA-S, and TA + E. The best performance is shown with an underline. Cora (low) means a low labeling rate setting, and Cora (high) denotes a high labeling rate setting.

	Cora (low)			Pubmed (low)		
	GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
TA	82.56 ± 0.73	81.62 ± 1.09	74.26 ± 0.93	82.63 ± 1.13	79.67 ± 0.80	80.38 ± 1.94
KEA-I + TA	83.20 ± 0.56	83.38 ± 0.63	74.34 ± 0.97	83.30 ± 1.75	81.16 ± 0.87	80.74 ± 2.44
KEA-S + TA	84.63 ± 0.58	85.02 ± 0.40	76.11 ± 2.66	82.93 ± 2.38	81.34 ± 1.51	80.74 ± 2.44
TA+E	83.38 ± 0.42	84.00 ± 0.09	75.73 ± 0.53	87.44 ± 0.49	86.71 ± 0.92	90.25 ± 1.56
	Cora (high)			**Pubmed (high)**		
	GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
TA	90.53 ± 2.33	89.10 ± 3.22	86.19 ± 4.38	89.65 ± 0.85	89.55 ± 1.16	91.39 ± 0.47
KEA-I + TA	91.12 ± 1.76	90.24 ± 2.93	87.88 ± 4.44	90.19 ± 0.83	90.60 ± 1.22	92.12 ± 0.74
KEA-S + TA	91.09 ± 1.78	92.30 ± 1.69	88.95 ± 4.96	90.40 ± 0.92	90.82 ± 1.30	91.78 ± 0.56
TA+E	90.68 ± 2.12	91.86 ± 1.36	87.00 ± 4.83	92.64 ± 1.00	93.35 ± 1.24	94.34 ± 0.86

APIs, it's not possible for us to test the whole dataset for these graphs. Considering the rate limit imposed by OpenAI[4], we randomly select 200 nodes from the test sets as our test data. In order to ensure that these 200 nodes better represent the performance of the entire set, we repeat all experiments twice. Additionally, we employ zero-shot performance as a sanity check, comparing it with the results in TAPE [22] to ensure minimal discrepancies.

We explore the following strategies:

1. **Zero-shot prompts**: This approach solely involves the attribute of a given node.
2. **Few-shot prompts**: On the basis of zero-shot prompts, few-shot prompts provide in-context learning samples together with their labels for LLMs to better understand the task. In addition to the node's content, this approach integrates the content and labels of randomly selected in-context samples from the training set. In the section, we adopt random sampling to select few-shot prompts.
3. **Zero-shot prompts with Chain-of-Thoughts (CoT)**: CoT [70] presents its effectiveness in various reasoning tasks, which can greatly improve LLMs' reasoning abilities. In this study, we test whether CoT can improve LLMs' capability on node classification tasks. On the basis of zero-shot prompts, we guide the LLMs to generate the thought process by using the prompt "think it step by step".
4. **Few-shot prompts with CoT**: Inspired by [82], which demonstrates that incorporating the CoT process generated by LLMs can further improve LLMs' reasoning

[4] https://platform.openai.com/docs/guides/rate-limits/overview

capabilities. Building upon the few-shot prompts, this approach enables the LLMs to generate a step-by-step thought process for the in-context samples. Subsequently, the generated CoT processes are inserted into the prompt as auxiliary information.

Output Parsing. In addition, we need a parser to extract the output from LLMs. We devise a straightforward approach to retrieve the predictions from the outputs. Initially, we instruct the LLMs to generate the results in a formatted output like "a python list". Then, we can use the symbols "[" and "]" to locate the expected outputs. It should be noted that this design aims to extract the information more easily but has little influence on the performance. We observe that sometimes LLMs will output contents that are slightly different from the expected format, for example, output the expected format "Information Retrieval" to "Information Extraction". In such cases, we compute the edit distance between the extracted output and the category names and select the one with the smallest distance. This method proves effective when the input context is relatively short. If this strategy encounters errors, we resort to extracting the first mentioned categories in the output texts as the predictions. If there's no match, then the model's prediction for the node is incorrect.

To reduce the variance of LLMs' predictions, we set the temperature to 0. For few-shot cases, we find that providing too much context will cause LLMs to generate outputs that are not compatible with the expected formats. Therefore, we set a maximum number of samples to ensure that LLMs generate outputs with valid formats. In this study, we choose this number to 2 and adopt accuracy as the performance metric.

5.1.1 Observations

Observation 11. LLMs present preliminary effectiveness on some datasets.

According to the results in Table 10, it is evident that LLMs demonstrate remarkable zero-shot performance on PUBMED. When it comes to OGBN-PRODUCTS, LLMs can achieve performance levels comparable to fine-tuned PLMs. However, there is a noticeable performance gap between LLMs and GNNs on CORA and PUBMED datasets. To gain a deeper understanding of this observation, it is essential to analyze the output of LLMs.

Observation 12. Wrong predictions made by LLMs are sometimes also reasonable.

After investigating the output of LLMs, we find that a part of the wrong predictions made by LLMs are very reasonable. An example is shown in Table 11. In this example, we can see that besides the ground truth label "Reinforcement Learning", "Neural Networks" is also a reasonable label, which also appears in the texts. We find that this is a common problem for CORA, CITESEER, and OGBN-ARXIV. For OGBN-ARXIV, there are usually multiple labels for one paper on the website. However, in the OGBN-ARXIV dataset, only one of them is chosen as the ground truth. This leads to a misalignment between LLMs' commonsense knowledge and the annotation bias inherent in these datasets. Moreover, we find that introducing few-shot samples presents little help to mitigate the annotation bias.

Observation 13. Chain-of-thoughts do not bring in performance gain.

For reasoning tasks in the general domain, chain-of-thoughts is believed to be an effective approach to increase LLM's reasoning capability [70]. However, we find that it's not effective for the node classification task. This phenomenon can be potentially explained by **Observation 12**. In contrast to mathematical reasoning, where a single answer is typically expected, multiple reasonable chains of thought can exist for node classification. An example is shown in Table 12. This phenomenon poses a challenge for LLMs as they may struggle to match the ground truth labels due to the presence of multiple reasonable labels.

Observation 14. For prompts that are very similar in semantics, there may be huge differences in their effects.

In addition, we observe that TAPE [22] implements a unique prompt on the OGBN-ARXIV dataset, yielding impressive results via zero-shot prompts. The primary distinction between their prompts and ours lies in the label design. Given that all papers originate from the computer science subcategory of Arxiv, they employ the brief term "arxiv cs subcategories" as a substitute for these 40 categories. Remarkably, this minor alteration contributes to a substantial enhancement in performance. To delve deeper into this phenomenon, we experiment with three disparate label designs: (1) Strategy 1: the original Arxiv identifier, such as "arxiv cs.CV"; (2) Strategy 2: natural language descriptors, like "computer vision"; and (3) Strategy 3: the specialized prompt, utilizing "arxiv cs subcategory" to denote all categories. Unexpectedly, we discover that Strategy 3 significantly outperforms the other two (refer to Table 13).

Given that LLMs undergo pre-training on extensive text corpora, it's likely that these corpora include papers from the Arxiv database. That specific prompt could potentially enhance the "activation" of these models' corresponding memory. However, the reason for the excellent results achieved by this kind of prompt might not stem from the simple data memorization of the LLM [25]. When applying to papers after 2023 that are not included in the pre-training corpus of the LLMs, this prompt also achieves similar effectiveness. This phenomenon reminds us that when using ChatGPT, sometimes providing more information in the prompt (such as category information from the OGBN-ARXIV dataset) may actually lead to a decrease in performance.

5.2 Incorporating Structural Information in the Prompts

As we note, LLMs can already present superior zero-shot performance on some datasets without providing any structural information. However, there is still a large performance gap between LLMs and GNNs in CORA, CITESEER, and OGBN-ARXIV. Then a question naturally raises that *whether we can further increase LLMs' performance by incorporating structural information?* To answer this problem, we first need to identify how to denote the structural information in the prompt. LLMs such as ChatGPT are not originally designed for graph structures, so they can not process adjacency matrices like GNNs. In this part, we study several ways to convey structural information and test their effectiveness on the CORA dataset.

Specifically, we first consider inputting the whole graph into the LLMs. Using CORA dataset as an example, we try to use prompts like "node 1: ⟨**paper content**⟩" to represent attributes, and prompts like "node 1 cites node 2" to represent the edge. However, we find that this approach is not

Table 10: Performance of LLMs on real-world text attributed graphs without structural information, we also include the result of GCN (or SAGE for OGBN-PRODUCTS) together with Sentence-BERT features. For CORA, CITESEER, PUBMED, we show the results of the low labeling rate setting.

	CORA	CITESEER	PUBMED	OGBN-ARXIV	OGBN-PRODUCTS
Zero-shot	67.00 ± 1.41	65.50 ± 3.53	90.75 ± 5.30	51.75 ± 3.89	70.75 ± 2.48
Few-shot	67.75 ± 3.53	66.00 ± 5.66	85.50 ± 2.80	50.25 ± 1.06	77.75 ± 1.06
Zero-shot with COT	64.00 ± 0.71	66.50 ± 2.82	86.25 ± 3.29	50.50 ± 1.41	71.25 ± 1.06
Few-shot with COT	64.00 ± 1.41	60.50 ± 4.94	85.50 ± 4.94	47.25 ± 2.47	73.25 ± 1.77
GCN/SAGE	82.20 ± 0.49	71.19 ± 1.10	81.01 ± 1.32	73.10 ± 0.25	82.51 ± 0.53

Table 11: A wrong but reasonable prediction made by LLMs

Paper: The Neural Network House: An overview; Typical home comfort systems utilize only rudimentary forms of energy management and conservation. The most sophisticated technology in common use today is an automatic setback thermostat. Tremendous potential remains for improving the efficiency of electric and gas usage...
Ground Truth: Reinforcement Learning
LLM's Prediction: Neural Networks

Table 12: An example that LLMs generate CoT processes not matching with ground truth labels

Paper: The Neural Network House: An overview.: Typical home comfort systems utilize only rudimentary forms of energy management and conservation. The most sophisticated technology in common use today is an automatic setback thermostat. Tremendous potential remains for improving the efficiency of electric and gas usage...
Generated Chain-of-thoughts: The paper discusses the use of neural networks for intelligent control and mentions the utilization of neural network reinforcement learning and prediction techniques. Therefore, the most likely category for this paper is 'Neural Networks'.
Ground Truth: Reinforcement Learning
LLM's Prediction: Neural Networks

Table 13: Performance of LLMs on OGB-Arxiv dataset, with three different label designs.

	Strategy 1	**Strategy 2**	**Strategy 3**
OGBN-ARXIV	48.5	51.8	74.5

feasible since LLMs usually present a small input context length restriction. As a result, we consider an "ego-graph" view, which refers to the subgraphs induced from the center nodes. In this way, we can narrow the number of nodes to be considered.

Specifically, we first organize the neighbors of the current nodes as a list of dictionaries consisting of attributes and labels of the neighboring nodes for training nodes. Then, the LLMs summarize the neighborhood information. It should be noted that we only consider 2-hop neighbors because GNNs typically have 2 layers, indicating that the 2-hop neighbor information is the most useful in most cases. Considering the input context limit of LLMs, we empirically find that each time we can summarize the attribute information of 5 neighbors. In this paper, we sample neighbors once and only summarize those selected neighbors. In practice, we can sample multiple times and summarize each of them to obtain more fine-grained neighborhood information.

Observation 15. Neighborhood summarization is likely to achieve performance gain.

From Table 14, we note that incorporating neighborhood information in either zero-shot or few-shot approaches yields performance gains compared to the zero-shot prompt without structural information except on the PUBMED dataset. By following the "homophily" assumption [87; 39], which suggests that neighboring nodes tend to share the same labels, the inclusion of neighboring information can potentially alleviate annotation bias. For instance, let's consider a paper from Arxiv covering general topics like transformers. Merely analyzing the content of this paper makes it difficult to determine which category the author would choose, as categories such as "Artificial Intelligence," "Machine Learning," and "Computer Vision" are all plausible options. However, by examining its citation relationships, we can better infer the author's bias. If the paper cites numerous sources from the "Computer Vision" domain, it is likely that the author is also a researcher in that field, thereby favoring the selection of this category. Consequently, structural information provides implicit supervision to assist LLMs in capturing the inherent annotation bias in the dataset. However, from the PUBMED dataset, we observe that incorporating neighborhood information results in clear performance drop, which necessitates a deep analysis below.

Observation 16. LLMs with structure prompts may suffer from heterophilous neighboring nodes.

From Table 14, we observe that LLMs perform worse on PUBMED after incorporating the structural information. To gain a deeper understanding, we focus on those nodes where zero-shot prompts without structural information can lead to correct prediction but prompts with 2-hop information can't.

An example of this kind of node is shown in Table 15. After analyzing the 2-hop neighbors of this node, we find that 15 out of 19 2-hop neighboring nodes have different labels against this node. This case is usually denoted as "heterophily" [87], which is a phenomenon in graph theory where nodes in a graph tend to connect with nodes that are dissimilar to them. In this case, we find that both GNNs and LLMs with a structure-aware prompt make wrong predictions. However, LLMs ignoring structural information get correct predictions, which indicates that LLMs with a structure-aware prompt may also suffer from the "het-

Table 14: Performance of LLMs on real-world text attributed graphs with summarized neighborhood information. For CORA, CITESEER, PUBMED, we show the results of the low labeling rate setting. We also include the result of GCN (or SAGE for OGBN-PRODUCTS) together with Sentence-BERT features.

	CORA	CITESEER	PUBMED	OGBN-ARXIV	OGBN-PRODUCTS
Zero-shot	67.00 ± 1.41	65.50 ± 3.53	90.75 ± 5.30	51.75 ± 3.89	70.75 ± 2.48
Few-shot	67.75 ± 3.53	66.00 ± 5.66	85.50 ± 2.80	50.25 ± 1.06	77.75 ± 1.06
Zero-Shot with 2-hop info	71.75 ± 0.35	62.00 ± 1.41	88.00 ± 1.41	55.00 ± 2.83	75.25 ± 3.53
Few-Shot with 2-hop info	74.00 ± 4.24	67.00 ± 4.94	79.25 ± 6.71	52.25 ± 3.18	76.00 ± 2.82
GCN/SAGE	82.20 ± 0.49	71.19 ± 1.10	81.01 ± 1.32	73.10 ± 0.25	82.51 ± 0.53

erophily" problem.

Table 15: GNNs and LLMs with structure-aware prompts are both wrong

Paper: Title: C-reactive protein and incident cardiovascular events among men with diabetes.
Abstract: OBJECTIVE: Several large prospective studies have shown that baseline levels of C-reactive protein (CRP) ...
Neighbor Summary: This paper focuses on different aspects of **type 2 diabetes** mellitus. It explores the levels of various markers such as tumor necrosis factor-alpha, interleukin-2 ...
Ground truth: "Diabetes Mellitus Type 1"
Structure-ignorant prompts: "Diabetes Mellitus Type 1"
Structure-aware prompt: "Diabetes Mellitus Type 2"
GNN: "Diabetes Mellitus Type 2"

5.3 Case Study: LLMs as the Pseudo Annotators

From Table 10, we show that LLMs can be good **zero-shot predictors** on several real-world graphs, which provides the possibility to conduct zero-shot inference on datasets without labels. Despite the effectiveness of LLMs, it still presents two problems: (1) The price of using LLMs' API is not cheap, and conducting inference on all testing nodes for large graphs incurs high costs; (2) Whether it is a locally deployed open-source LLM or a closed source LLM accessed through an API, the inference with these LLMs are much slower than GNNs, since the former has high computational resource requirements, while the latter has rate limits. One potential solution to these challenges is leveraging the knowledge of LLMs to train smaller models like GNNs, which inspires a potential application of LLMs to be used as annotators.

Based on the preliminary experimental outcomes, LLMs display encouraging results on certain datasets, thus highlighting their potential for generating high-quality pseudo-labels. However, the use of LLMs as an annotator introduces a new challenge. A key consideration lies in deciding the nodes that should be annotated. Unlike the self-labeling in GNNs[8; 34; 32], where confidence-based or information-based metrics are employed to estimate the quality of pseudo-labels. It remains a difficult task to determine the confidence of pseudo-labels generated by LLMs. Additionally, different nodes within a graph have distinct impacts on other nodes [72]. Annotating certain nodes can result in a more significant performance improvement compared to others. Consequently, the primary challenge can be summarized as follows: how can we effectively select both the critical nodes within the graph and the reliable nodes in the context of LLMs?

Taking into account the complexity of these two challenges, we don't intend to comprehensively address them in this paper. Instead, we present a preliminary study to evaluate the performance of a simple strategy: randomly selecting a subset of nodes for annotation. It is worth noting that advanced selection strategies such as active learning [72] could be adopted to improve the final performance. We leave such exploration as future work. Regarding the annotation budget, we adopt a "low labeling rate" setting, wherein we randomly select a total of 20 nodes multiplied by the number of classes. For the selected nodes, we adopt 75% of them as training nodes and the rest as validation nodes. Consequently, we annotate a total of 140 nodes in the CORA dataset and 60 nodes in the PUBMED dataset. In this part, we use GCN as the GNN model and adopt the embeddings generated by the Sentence-BERT model. The results are shown in Table 16. We can observe that training GCN on the pseudo labels can lead to satisfying performance. Particularly, it can match the performance of GCN trained on ground truth labels with 10 shots per class. As a reference, around 67% of the pseudo labels for CORA can match ground truth labels, while around 93% of the pseudo labels for PUBMED are ground truth labels.

Table 16: Performance of GCN trained on either pseudo labels generated by LLMs, or ground truth labels

	Cora	**Pubmed**
Using pseudo labels		
20 shots × #class	64.95 ± 0.98	71.70 ± 1.06
Using ground truth		
3 shots per class	52.63 ± 1.46	59.35 ± 2.67
5 shots per class	58.97 ± 1.41	65.98 ± 0.74
10 shots per class	69.87 ± 2.27	71.51 ± 0.77

Observation 17. The quality of pseudo labels is key to downstream performance.

Although we don't place significant emphasis on the selection of nodes to be labeled, the preliminary results show that there is relatively little variance among different random selections. Comparing this to the impact of pseudo labels, we observe that the quality of pseudo labels can make a significant difference. When higher quality pseudo labels are used, GNNs perform much better on PUBMED compared to CORA. This result highlights the importance of developing an approach to select confident nodes for LLMs.

Observation 18. Getting the confidence by simply

prompting the LLMs may not work since they are too "confident".
Based on previous observations, we check some simple strategies to achieve the confidence level of LLMs' outputs. Initially, we attempt to prompt the LLMs directly for their confidence level. However, we discover that most of the time, LLMs simply output a value of 1, rendering it meaningless. Examples are shown in Table 17.

Table 17: Prompts used to generate neighbor summary

Instruction
Output the confidence level in the range of 0 to 1 and the most 1 possible category of this paper as a python dict, like "prediction": "XX", "confidence": "XX"

Another potential solution is to utilize LLMs that support prediction logits, such as text-davinci-003. However, we observe that the probability of the outputs from these models is consistently close to 1, rendering the output not helpful.

5.4 Case Study: Applying LLMs to handle out-of-distribution data

Out-of-distribution (OOD) learning addresses scenarios where training and test data are drawn from different distributions. Given the ubiquity of distribution shifts in graph data [29], OOD generalization on graphs has emerged as a crucial research direction in recent years. A recent benchmark, GOOD [17], reveals that existing GNN-based models struggle with robustness when confronted with distributional shifts. In contrast, LLMs have demonstrated commendable robustness on textual data in the presence of OOD scenarios [67]. Node classification on the TAG, when disregarding graph structures, can also be considered as a text classification task. Therefore, in this section, we initiate a preliminary exploration into the application of LLMs for OOD scenarios on graphs.

Experimental Setups. We adopt the GOOD-Arxiv dataset from the GOOD benchmark [17] considering its text attribute availability. Specifically, we adopt all four types of the OOD shift: "Concept-degree", "Covariate-degree", "Concept-time", and "Covariate-time" from the GOOD. The final results are shown in Table 18. We adopt the prompt from TAPE [22] since it achieves better performance on the Ogbn-arxiv dataset. For comparison, we take the best baseline models from the GOOD benchmark.

Table 18: OOD performance comparison. "Val" means the results on the IID validation sets. "Test" indicates the results of the OOD test sets. We can see that LLMs-as-Predictors consistently outperform the best GNN-based OOD baselines. Moreover, the gap between IID performance and OOD performance is small.

	Val	Test	Best baseline (test)
concept degree	73.01	72.79	63.00
covariate degree	70.23	68.21	59.08
concept time	72.66	71.98	67.45
covariate time	74.28	74.37	71.34

Observation 19. LLMs-as-Predictors demonstrate robustness when facing OOD data.
From Table 18, we find that LLMs-as-Predictors present promising robustness against OOD data. It should be noted that we only try a simple structure-ignorant prompt, and we may further improve the OOD performance of LLMs by selecting proper in-context samples and incorporating structural information. In a nutshell, LLMs present great potential to enhance the OOD generalization capability of graph models.

6. RELATED WORK

Following our proposed two pipelines, i.e., LLMs as the Enhancers and LLMs as the Predictors, we review existing works in this section.

6.1 LLMs as the Enhancers

In the recent surge of research, increasing attention has been paid on the intersection of LLMs and GNNs in the realm of TAGs [83; 6; 78; 77; 49; 22; 86; 24; 33; 10]. Compared to shallow embeddings, LLMs can provide a richer repository of commonsense knowledge, which could potentially enhance the performance of downstream tasks [51].

Several studies employ PLMs as text encoders, transforming text attributes into node features, which can thus be classified as ***feature-level enhancement***. The integration structures vary among these works: some adopt a simple cascading structure [49; 6; 78; 37], while others opt for an iterative structure [83; 74; 77]. For those utilizing the cascading structure, preliminary investigations have been conducted to determine how the quality of text embeddings affects downstream classification performance [49]. GIANT [6] attempts to incorporate structural information into the pre-training stage of PLMs, achieving improved performance albeit with additional training overhead. SimTEG [10] suggests that using embeddings obtained through efficiently fine-tuned parameters to replace the original embeddings of pre-trained language models can solve the problem of overfitting during fine-tuning, thereby further enhancing the performance of the cascading structure. OneForAll [33] further adopts sentence embedding model to unify the feature space, and propose a unified model for diverse tasks across multiple datasets. This cascading structure has also been successfully applied to tasks such as fact verification [37] and question answering [78]. However, despite its simplicity, recent studies [83] have identified potential drawbacks of the cascading structure. Specifically, it establishes a tenuous connection between the text attribute and the graph. The embeddings generated by the PLMs do not take graph structures into account, and the parameters of the PLMs remain constant during the GNN training process. Alternatively, in the iterative structure, Graphformers [74] facilitates the co-training of PLMs and GNNs using each other's generated embeddings. GLEM [83] takes this a step further by considering pseudo labels generated by both PLMs and GNNs and incorporating them into the optimization process. DRAGON [77] successfully extends the iterative structure to the knowledge graph domain.

Compared to these studies focusing on PLMs, a recent study [22] considers the usage of embedding-invisible LLMs such as ChatGPT [45] for representation learning on TAGs, which aims to adopt LLMs to enhance the text attributes and thus can be categorized into ***text-level enhancement***. This

work introduces a prompt designed to generate explanations for the predictions made by LLMs. These generated explanations are subsequently encoded into augmented features by PLMs. Through the ensemble of these augmented features with the original features, the proposed methodology demonstrates its efficacy and accomplishes state-of-the-art performance on the OGBN-ARXIV leaderboard [23]. Nevertheless, the study offers limited analytical insights into the underlying reasons for the success of this approach. Additionally, we have identified a potential concern regarding the prompts utilized in the referenced study.

Another work pertaining to the integration of LLMs and GNNs is the Graph-Toolformer [80]. Drawing inspirations from Toolformer [56], this study utilizes LLMs as an interface to bridge the natural language commands and GNNs. This approach doesn't change the features and training of GNNs, which is out of our scope.

6.2 LLMs as the Predictors

While *LLMs-as-Enhancers* have proven to be effective, the pipeline still requires GNNs for final predictions. In a significant shift from this approach, recent studies [18; 65] have begun exploring a unique pipeline that solely relies on LLMs for final predictions. These works fall under the category of *LLMs-as-Predictors*. The first series of work focus on applying closed-source LLMs without tuning the parameters. GPT4Graph [18] evaluates the potential of LLMs in executing knowledge graph (KG) reasoning and node classification tasks. Their findings indicate that these models can deliver competitive results for short-range KG reasoning but struggle with long-range KG reasoning and node classification tasks. However, its presentation is pretty vague and they don't give the detailed format of the prompt they use. Considering the publicity of the Arxiv data, the data leakage problem in evaluation is further studied in [25]. NLGraph [65] introduces a synthetic benchmark to assess graph structure reasoning capabilities. The study primarily concentrates on traditional graph reasoning tasks such as shortest path, maximum flow, and bipartite matching, while only offering limited analysis on node classification tasks. This does not align with our central focus, primarily on graph learning, with a specific emphasis on node classification tasks. GraphText [84] further tries to apply LLMs to a broader range of non-text-attributed graphs by converting the original features into clustering centers or pseudo labels. LLM4Dyg [81] further evaluates LLMs' capability for temporal graph-related tasks. LLMGNN [4] and GPT4GNAS [66] apply LLMs-as-predictors as annotators and agents for neural architecture search, respectively.

As these closed-source LLMs only accept text-type inputs, the first type of methods requires transforming graphs into certain form of natural language, either directly using node attributes or describing the graph structure using natural language. Meanwhile, due to the input length limitations of LLMs, this transformation process often results in the loss of a considerable amount of information from the graph. Therefore, the second type of work involves fine-tuning LLMs to enable them to understand graph information represented as embeddings. InstructGLM [79] combines textual instructions with node features in embedding form, enabling LLMs to understand node features through instruction tuning. Subsequently, it predicts the type of nodes based on the given instructions. GraphGPT [62] further introduces cross-modal contrastive learning to align the graph and text feature spaces. It also introduces dual-stage instruction tuning, where the first stage adopts self-supervised instruction tuning to make LLMs better understand graph-structured information. The second stage adopts task-specific fine-tuning to allow LLMs achieve task-specific knowledge and then make predictions. GraphLLM [3] and DGTL [50] apply this pipeline to graph reasoning tasks and graph representation learning.

7. CONCLUSIONS, LIMITATIONS, AND FUTURE DIRECTIONS

In this section, we summarize our key findings, present the limitations of this study and discuss the potential directions of leveraging LLMs in graph machine learning.

7.1 Key Findings

In this paper, we propose two potential pipelines: *LLMs-as-Enhancers* and *LLMs-as-Predictors* that incorporate LLMs to handle the text-attributed graphs. Our rigorous empirical studies reveal several interesting findings which provide new insights for future studies. We highlight some key findings below and more can be found from Observation 1 to Observation 19.

Finding 1. For *LLMs-as-Enhancers*, deep sentence embedding models present effectiveness in terms of performance and efficiency. We empirically find that when we adopt deep sentence embedding models as enhancers at the feature level, they present good performance under different dataset split settings, and also scalability. This indicates that they are good candidates to enhance text attributes at the feature level.

Finding 2. For *LLMs-as-Enhancers*, the combination of LLMs' augmentations and ensembling demonstrates its effectiveness. As demonstrated in Section 4.2, when LLMs are utilized as enhancers at the text level, we observe performance improvements by ensembling the augmented attributes with the original attributes across datasets and data splits. This suggests a promising approach to enhance the performance of attribute-related tasks. The proposed pipeline involves augmenting the attributes with LLMs and subsequently ensembling the original attributes with the augmented ones.

Finding 3. For *LLMs-as-Predictors*, LLMs present preliminary effectiveness but also indicate potential evaluation problem. In Section 5, we conduct preliminary experiments on applying LLMs as predictors, utilizing both textual attributes and edge relationships. The results demonstrate that LLMs present effectiveness in processing textual attributes and achieving good zero-shot performance on certain datasets. Moreover, our analysis reveals two potential problems within the existing evaluation framework: (1) There are instances where LLMs' inaccurate predictions can also be considered reasonable, particularly in the case of citation datasets where multiple labels may be appropriate. (2) We find a potential test data leakage problem on OGBN-ARXIV, which underscores the need for a careful reconsideration of how to appropriately evaluate the performance of LLMs on real-world datasets.

7.2 Limitations

A deeper understanding of the effectiveness of text embeddings. Despite the effectiveness of deep sentence

embedding models, our understanding of why their embeddings outperform PLMs' on node classification tasks remains limited. Furthermore, we observe a performance gap between deep sentence embedding models and GLEM on the OGBN-PRODUCTS dataset, which may be related to the domains of the dataset. Moreover, as shown in Observation 4, GNNs demonstrate different levels of effectiveness on different text embeddings. However, we give limited explanations for this phenomenon. To gain a deeper understanding, we need to have a look at the original feature space and the feature space after aggregation. This phenomenon may potentially be related to the anistrophy in language model embeddings [12]. More in-depth analysis is required to better understand these phenomena.

Costs of LLM augmentations. In the work, we study TAPE and KEA to enhance the textual attributes at the text level. Although these methods have proven to be effective, they require querying LLMs' APIs at least N times for a graph with N nodes. Given the cost associated with LLMs, this poses a significant expense when dealing with large-scale datasets. Consequently, we have not presented results for the OGBN-ARXIV and OGBN-PRODUCTS datasets.

Text-formatted hand-crafted prompts to represent graphs. In Section 5, we limit our study to the use of "natural language" prompts for graph representation. However, various other formats exist for representing graphs in natural language such as XML, YAML, GML, and more [55]. Moreover, we mainly design these prompts in a hand-crafted way, which is mainly based on trial and error. It's thus worthwhile to consider exploring more prompt formats and how to come up with automatic prompts.

7.3 Future Directions

Extending the current pipelines to more tasks and more types of graphs. In this study, our primary focus is on investigating the node classification task for text-attributed graphs. Nevertheless, it remains unexplored whether these two pipelines can be extended to other graph-learning tasks or other types of graphs. Certain tasks necessitate the utilization of long-range information [11], and representing such information within LLMs' limited input context poses a significant challenge. Furthermore, we demonstrate that LLMs exhibit promising initial results in graphs containing abundant textual information, particularly in natural language. However, the exploration of their effective extension to other types of graphs with non-natural language information, such as molecular graph [13; 30], still needs further exploration.

Using LLMs more efficiently. Despite the effectiveness of LLMs, the inherent operational efficiency and operational cost of these models still pose significant challenges. Taking ChatGPT, which is accessed through an API, as an example, the current billing model incurs high costs for processing large-scale graphs. As for locally deployed open-source large models, even just using them for inference requires substantial hardware resources, not to mention training the models with parameter updates. Therefore, developing more efficient strategies to utilize LLMs is currently a challenge.

Evaluating LLMs' capability for graph learning tasks. In this paper, we briefly talk about the potential pitfalls of the current evaluation framework. There are mainly two problems: (1) the test data may already appear in the training corpus of LLMs, which is referred to as "contamination" [5] (2) the ground truth labels may present ambiguity, and the performance calculated based on them may not reflect LLMs' genuine capability. For the first problem, one possible mitigation is to use the latest dataset which is not included in the training corpus of LLMs. However, that means we need to keep collecting data and annotating them, which seems not an effective solution. For the second problem, one possible solution is to reconsider the ground truth design. For instance, for the categorization of academic papers, we may adopt a multi-label setting and select all applicable categories as the ground truth. However, for more general tasks, it remains a challenge to design more reasonable ground truths. Generally speaking, it's a valuable future direction to rethink how to properly evaluate LLMs.

Aligning the feature space of graph models and LLMs. Currently, a major obstacle hindering the wider application of LLMs in the field of graph learning is the discrepancy between the feature space of LLMs and that of graphs. This discrepancy makes it difficult for LLMs to effectively understand information in the graph domain. There are mainly two approaches to address this issue in current work. The first approach is to translate the information on the graph into natural language that LLMs can understand. The second approach involves directly inputting the graph information in the form of embeddings and then using instruction tuning to enable LLMs to understand this information. However, both methods have their evident limitations. For the first method, the translation process can result in information loss, and the inherent input length limitation of LLMs also prevents users from inputting large-scale graphs. For the second method, the introduction of tuning significantly increases computational overhead. Is there a better way to align LLMs with graphs? A recent work targeting multimodality [47] has shown new possibilities. It demonstrates that with fixed LLM parameters, only a linear transformation layer is needed to convert information from the visual domain into content that can be effectively processed by LLMs, and such an architecture also holds great potential in the field of graph machine learning.

8. REFERENCES

[1] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[2] S. Bubeck, V. Chandrasekaran, R. Eldan, J. A. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y.-F. Li, S. M. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv*, abs/2303.12712, 2023.

[3] Z. Chai, T. Zhang, L. Wu, K. Han, X. Hu, X. Huang, and Y. Yang. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*, 2023.

[4] Z. Chen, H. Mao, H. Wen, H. Han, W. Jin, H. Zhang, H. Liu, and J. Tang. Label-free node classification on graphs with large language models (llms). *arXiv preprint arXiv:2310.04668*, 2023.

[5] `https://hitz-zentroa.github.io/lm-contamination/`

[5] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[6] E. Chien, W.-C. Chang, C.-J. Hsieh, H.-F. Yu, J. Zhang, O. Milenkovic, and I. S. Dhillon. Node feature extraction by self-supervised multi-scale neighborhood prediction. In *ICLR 2022*, 2022.

[7] A. Creswell, M. Shanahan, and I. Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023.

[8] E. Dai, C. Aggarwal, and S. Wang. Nrgnn: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 227–236, New York, NY, USA, 2021. Association for Computing Machinery.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[10] K. Duan, Q. Liu, T.-S. Chua, S. Yan, W. T. Ooi, Q. Xie, and J. He. Simteg: A frustratingly simple approach improves textual graph learning. *arXiv preprint arXiv:2308.02565*, 2023.

[11] V. P. Dwivedi, L. Rampášek, M. Galkin, A. Parviz, G. Wolf, A. T. Luu, and D. Beaini. Long range graph benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[12] K. Ethayarajh. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[13] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric. *ArXiv*, abs/1903.02428, 2019.

[14] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, and J. Zhang. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *ArXiv*, abs/2303.14524, 2023.

[15] C. L. Giles, K. D. Bollacker, and S. Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, DL ́98, pages 89–98, New York, NY, USA, 1998. ACM.

[16] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. *ArXiv*, abs/1704.01212, 2017.

[17] S. Gui, X. Li, L. Wang, and S. Ji. GOOD: A graph out-of-distribution benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[18] J. Guo, L. Du, and H. Liu. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*, 2023.

[19] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[20] Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[21] P. He, X. Liu, J. Gao, and W. Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[22] X. He, X. Bresson, T. Laurent, and B. Hooi. Explanations as features: Llm-based features for text-attributed graphs. *arXiv preprint arXiv:2305.19523*, 2023.

[23] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[24] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun. Gpt-gnn: Generative pre-training of graph neural networks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[25] J. Huang, X. Zhang, Q. Mei, and J. Ma. Can llms effectively leverage graph structural information: When and why. *arXiv preprint arXiv:2309.16595*, 2023.

[26] Y. Ji, Y. Gong, Y. Peng, C. Ni, P. Sun, D. Pan, B. Ma, and X. Li. Exploring chatgpt's ability to rank content: A preliminary study on consistency with human preferences. *ArXiv*, abs/2303.07610, 2023.

[27] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[28] G. Li, M. Müller, B. Ghanem, and V. Koltun. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pages 6437–6449. PMLR, 2021.

[29] H. Li, X. Wang, Z. Zhang, and W. Zhu. Out-of-distribution generalization on graphs: A survey. *arXiv preprint arXiv:2202.07987*, 2022.

[30] J. Li, Y. Liu, W. Fan, X. Wei, H. Liu, J. Tang, and Q. Li. Empowering molecule discovery for molecule-caption translation with large language models: A chatgpt perspective. *ArXiv*, abs/2306.06615, 2023.

[31] Q. Li, X. Li, L. Chen, and D. Wu. Distilling knowledge on text graph for social media attribute inference. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2024–2028, New York, NY, USA, 2022. Association for Computing Machinery.

[32] Y. Li, J. Yin, and L. Chen. Informative pseudo-labeling for graph neural networks with few labels. *Data Mining and Knowledge Discovery*, 37(1):228–254, 2023.

[33] H. Liu, J. Feng, L. Kong, N. Liang, D. Tao, Y. Chen, and M. Zhang. One for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149*, 2023.

[34] H. Liu, B. Hu, X. Wang, C. Shi, Z. Zhang, and J. Zhou. Confidence may cheat: Self-training on graph neural networks under distribution shift. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 1248–1258, New York, NY, USA, 2022. Association for Computing Machinery.

[35] J. Liu, C. Liu, R. Lv, K. Zhou, and Y. Zhang. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149*, 2023.

[36] W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, and P. Wang. K-bert: Enabling language representation with knowledge graph. In *AAAI Conference on Artificial Intelligence*, 2019.

[37] Z. Liu, C. Xiong, M. Sun, and Z. Liu. Fine-grained fact verification with kernel graph attention network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7342–7351, Online, July 2020. Association for Computational Linguistics.

[38] Y. Ma and J. Tang. *Deep Learning on Graphs*. Cambridge University Press, 2021.

[39] H. Mao, Z. Chen, W. Jin, H. Han, Y. Ma, T. Zhao, N. Shah, and J. Tang. Demystifying structural disparity in graph neural networks: Can one size fit all? *arXiv preprint arXiv:2306.01323*, 2023.

[40] A. McCallum, K. Nigam, J. D. M. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[41] A. Miaschi and F. Dell'Orletta. Contextual and non-contextual word embeddings: an in-depth linguistic investigation. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 110–119, Online, July 2020. Association for Computational Linguistics.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[43] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[44] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. A. Tezak, J. W. Kim, C. Hallacy, J. Heidecke, P. Shyam, B. Power, T. E. Nekoul, G. Sastry, G. Krueger, D. P. Schnurr, F. P. Such, K. S.-K. Hsu, M. Thompson, T. Khan, T. Sherbakov, J. Jang, P. Welinder, and L. Weng. Text and code embeddings by contrastive pre-training. *ArXiv*, abs/2201.10005, 2022.

[45] OpenAI. Introducing chatgpt, 2022.

[46] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[47] Z. Pang, Z. Xie, Y. Man, and Y.-X. Wang. Frozen transformers in language models are effective visual encoder layers. *arXiv preprint arXiv:2310.12973*, 2023.

[48] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel. Language models as knowledge bases? *ArXiv*, abs/1909.01066, 2019.

[49] S. Purchase, A. Zhao, and R. D. Mullins. Revisiting embeddings for graph neural networks. *ArXiv*, abs/2209.09338, 2022.

[50] Y. Qin, X. Wang, Z. Zhang, and W. Zhu. Disentangled representation learning with large language models for text-attributed graphs. *arXiv preprint arXiv:2310.18152*, 2023.

[51] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63:1872 – 1897, 2020.

[52] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[53] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[54] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[55] M. Roughan and S. J. Tuke. Unravelling graph-exchange file formats. *ArXiv*, abs/1503.02781, 2015.

[56] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[57] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, Sep. 2008.

[58] C. Sun, H. Gu, and J. Hu. Scalable and adaptive graph neural networks with self-label-enhanced training. *arXiv preprint arXiv:2104.09376*, 2021.

[59] T. Sun, Y. Shao, H. Qian, X. Huang, and X. Qiu. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pages 20841–20855. PMLR, 2022.

[60] X. Sun, X. Li, J. Li, F. Wu, S. Guo, T. Zhang, and G. Wang. Text classification via large language models. *ArXiv*, abs/2305.08377, 2023.

[61] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu. Ernie: Enhanced representation through knowledge integration. *ArXiv*, abs/1904.09223, 2019.

[62] J. Tang, Y. Yang, W. Wei, L. Shi, L. Su, S. Cheng, D. Yin, and C. Huang. Graphgpt: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023*, 2023.

[63] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[64] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[65] H. Wang, S. Feng, T. He, Z. Tan, X. Han, and Y. Tsvetkov. Can language models solve graph problems in natural language? *arXiv preprint arXiv:2305.10037*, 2023.

[66] H. Wang, Y. Gao, X. Zheng, P. Zhang, H. Chen, and J. Bu. Graph neural architecture search with gpt-4. *arXiv preprint arXiv:2310.01436*, 2023.

[67] J. Wang, X. Hu, W. Hou, H. Chen, R. Zheng, Y. Wang, L. Yang, H. Huang, W. Ye, X. Geng, et al. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. *arXiv preprint arXiv:2302.12095*, 2023.

[68] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[69] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. J. Zhao, J. Li, A. Smola, and Z. Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs. *ArXiv*, abs/1909.01315, 2019.

[70] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[71] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[72] Y. Wu, Y. Xu, A. Singh, Y. Yang, and A. W. Dubrawski. Active learning for graph neural networks via node feature propagation. *ArXiv*, abs/1910.07567, 2019.

[73] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2:109–127, 2021.

[74] J. Yang, Z. Liu, S. Xiao, C. Li, D. Lian, S. Agrawal, A. S, G. Sun, and X. Xie. Graphformers: GNN-nested transformers for representation learning on textual graph. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[75] Z. Yang, W. W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *ArXiv*, abs/1603.08861, 2016.

[76] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. *ArXiv*, abs/1809.05679, 2018.

[77] M. Yasunaga, A. Bosselut, H. Ren, X. Zhang, C. D. Manning, P. Liang, and J. Leskovec. Deep bidirectional language-knowledge graph pretraining. In *Neural Information Processing Systems (NeurIPS)*, 2022.

[78] M. Yasunaga, J. Leskovec, and P. Liang. Linkbert: Pretraining language models with document links. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8003–8016, 2022.

[79] R. Ye, C. Zhang, R. Wang, S. Xu, and Y. Zhang. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*, 2023.

[80] J. Zhang. Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt. *arXiv preprint arXiv:2304.11116*, 2023.

[81] Z. Zhang, X. Wang, Z. Zhang, H. Li, Y. Qin, S. Wu, and W. Zhu. Llm4dyg: Can large language models solve problems on dynamic graphs? *arXiv preprint arXiv:2310.17110*, 2023.

[82] Z. Zhang, A. Zhang, M. Li, and A. J. Smola. Automatic chain of thought prompting in large language models. *ArXiv*, abs/2210.03493, 2022.

[83] J. Zhao, M. Qu, C. Li, H. Yan, Q. Liu, R. Li, X. Xie, and J. Tang. Learning on large-scale text-attributed graphs via variational inference. In *The Eleventh International Conference on Learning Representations*, 2023.

[84] J. Zhao, L. Zhuo, Y. Shen, M. Qu, K. Liu, M. Bronstein, Z. Zhu, and J. Tang. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*, 2023.

[85] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J. Nie, and J. rong Wen. A survey of large language models. *ArXiv*, abs/2303.18223, 2023.

[86] J. Zhu, Y. Cui, Y. Liu, H. Sun, X. Li, M. Pelger, L. Zhang, T. Yan, R. Zhang, and H. Zhao. Textgnn: Improving text encoder via graph neural network in sponsored search. *Proceedings of the Web Conference 2021*, 2021.

[87] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7793–7804. Curran Associates, Inc., 2020.

APPENDIX

A. DATASETS

In this work, we mainly use the following five real-world graph datasets. Their statistics are shown in Table 19.

Table 19: Statistics of the graph datasets.

Dataset	#Nodes	#Edges	Task	Metric
CORA [40]	2,708	5,429	7-class classif.	Accuracy
CITESEER * [15]	3,186	4,277	6-class classif.	Accuracy
PUBMED [57]	19,717	44,338	3-class classif.	Accuracy
OGBN-ARXIV [23]	169,343	1,166,243	40-class classif.	Accuracy
OGBN-PRODUCTS [23]	2,449,029	61,859,140	47-class classif.	Accuracy

A.1 Dataset Description

In this part, we give a brief introduction to each graph dataset. It should be noted that it's cumbersome to get the raw text attributes for some datasets, and we will elaborate them below. The structural information and label information of these datasets can be achieved from Pyg [6]. We will also release the pre-processed versions of these datasets to assist future related studies.

Cora [40] CORA is a paper citation dataset with the following seven categories: ['Rule Learning', 'Neural Networks', 'Case Based', 'Genetic Algorithms', 'Theory', 'Reinforcement Learning', 'Probabilistic Methods']. The raw text attributes can be obtained from `https://people.cs.umass.edu/~mccallum/data.html`.

Citeseer [15] CITESEER is a paper citation dataset with the following seven categories: ["Agents", "ML", "IR", "DB", "HCI", "AI"]. Note that we find that the TAG versopm only contains the text attributes for 3186 nodes. As a result, we take the graph consisted of these 3186 nodes with 4277 edges.

Pubmed [57] PUBMED is a paper citation dataset consisting scientific journals collected from the PubMed database with the following three categories: ['Diabetes Mellitus, Experimental', 'Diabetes Mellitus Type 1', 'Diabetes Mellitus Type 2'].

Ogbn-arxiv and Ogbn-products [23] These dataset are selected from the popular OGB benchmark [23], and descriptions for these datasets can be found in `https://ogb.stanford.edu/docs/nodeprop`.

B. EXPERIMENT SETUPS

B.1 Computing Environment

We implement all the baseline models with PyG [13], DGL [69], and transformers [71] modules. The experiments were conducted in a GPU server with eight NVIDIA RTX A5000 GPUs, each with 24GB VRAM.

B.2 Hyperparameters

For RevGAT, GraphSage, and SAGN models, we directly adopt the best hyperparameters from the OGB leaderboard [7]. For Deberta-base on CORA and PUBMED, we follow the hyperparameter setting of TAPE [22]. In terms of GLEM, for the LM part, we follow the hyperparameter setting in their reporsitory [8]. For GCN, GAT, MLP, we use the following hyperparameter search range.

(a) **Hidden dimension:** {8, 16, 32, 64, 128, 256}.
(b) **Number of layers:** {1, 2, 3}
(c) **Normalization:** {None, BatchNorm};
(d) **Learning rate:** {1e-2, 5e-2, 5e-3, 1e-3}
(e) **Weight Decay:** {1e-5, 5e-5, 5e-4, 0}
(f) **Dropout:** {0., 0.1, 0.5, 0.8}
(g) **Number of heads for GAT:** {1, 4, 8}

C. DEMONSTRATIONS OF TAPE

Examples for Pubmed After analyzing the PUBMED dataset, we find an interesting phenomenon that sometimes the label of the paper just appears in the raw text attributes. An example is shown in Table 20. This property of PUBMED may be related to the superior zero-shot performance of LLMs on this dataset. This can also potentially explain why GCN and GAT are outperformed by MLP in the high labeling ratio. When the link between node attributes and node labels can be easily found and adequate to determine the categories, incorporating neighbors coming from other categories will introduce noise.

Table 20: An illustrative example for PUBMED

Title: Predictive power of sequential measures of albuminuria for progression to ESRD or death in Pima Indians with **type 2 diabetes**.
... (content omitted here)
Ground truth label: Diabetes Mellitus Type 2

[6] `https://pytorch-geometric.readthedocs.io/en/latest/modules/data.html`

[7] `https://github.com/snap-stanford/ogb`

[8] `https://github.com/AndyJZhao/GLEM`