Characterizing Learning Difficulty in Graph-structured Data:

An Empirical Study of Models and Data

Tianqi Zhao

Characterizing Learning Difficulty in Graph-Structured Data:

An Empirical Study of Models and Data

Characterizing Learning Difficulty in Graph-Structured Data:

An Empirical Study of Models and Data

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus,
Prof. dr. ir. H. Bijl,
chair of the Board for Doctorates
to be defended publicly on
Tuesday, 12 May 2026 at 10:00

by

Tianqi Zhao

This dissertation has been approved by the (co)promotors:

promotor: Prof. dr. A. Hanjalic
copromotor: Dr. ing. M. Khosla

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. A. Hanjalic	Delft University of Technology, promotor
Dr. ing. M. Khosla	Delft University of Technology, copromotor
Independent Members:	
Prof. dr. G.J.P.M. Houben	Delft University of Technology
Prof. dr. I.R. van de Poel	Delft University of Technology
Prof. dr. M. Cochez	Abo Akademi University, Finland / Vrije Universiteit Amsterdam
Prof. dr. J. Yin	The University of Sydney, Australia
Dr. S. Yu	Vrije Universiteit Amsterdam

Keywords: Graph Neural Networks, Multi-label Node Classification, Continual Graph Learning, Instance-level Node Profiling
Printed by: ProefschriftMaken

ISBN 978-94-6518-037-3

An electronic version of this dissertation is available at http://repository.tudelft.nl/.

The days when you are stuck may be the days when a new landscape reveals itself.

Shi Tiesheng

PROPOSITIONS

accompanying the dissertation
Characterizing Learning Difficulty in Graph-Structured Data: An Empirical Study of Models and Data
by
Tianqi Zhao

1. Existing GNNs for single-label static graphs are not suitable for multi-label ones. *(This proposition pertains to this dissertation.)*
2. In graph continual learning, multi-label graphs should not be treated as a simple extension of single-label graphs. *(This proposition pertains to this dissertation.)*
3. Research progress on multi-label node classification suffers from insufficient understanding of simple baseline approaches. *(This proposition pertains to this dissertation.)*
4. Current evaluation metrics for graph neural networks fail to capture node-level behavioral differences. *(This proposition pertains to this dissertation.)*
5. Peer pressure in academia endangers scientific progress.
6. PhD theses would benefit from a section on unsuccessful research directions.
7. Accessing the modules of doctoral education program on a first-come-first-served basis goes against the objectives of doctoral education.
8. A PhD thesis should be composed of concise papers.
9. The rapidly growing size of computer science conferences endangers scientific quality, discussion and engagement.
10. Knowing when to stop is as important as knowing where to start in research.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotor Prof. Dr. Alan Hanjalic and copromotor Dr. ing. Megha Khosla.

CONTENTS

SUMMARY

Graphs provide a natural way to represent both the attributes of individual entities and the structure of their interconnections, making them a powerful framework for modeling complex systems with intricate relationships. Such networks appear across diverse domains, including social networks, biology, quantum physics, and knowledge graphs. For example, proteins and their interactions in protein–protein networks, or users and friendships in social networks.

In recent years, Graph Neural Networks (GNNs) have become the dominant paradigm for learning from graph-structured data, achieving strong results in tasks such as node classification, link prediction, and graph classification on academic benchmark datasets. However, a closer examination of one of the real-world scenario, multi-label node classification (MLNC) datasets, reveals more complicated attribute distributions and substantial data quality issues where GNNs fail to learn. Many real-world graphs are noisy, exhibit unbalanced label distributions, and display low label homophily, which complicates the extraction of meaningful information from local neighborhoods. This observation raises an important question: to what extent do the structural and distributional properties of graph data shape the performance of the model?

To investigate this, existing multi-label graph datasets were systematically analyzed, and a synthetic graph generator with tunable parameters was developed to enable controlled exploration of key factors such as homophily, degree distribution, and label assignment. With this foundation, a subsequent question emerged: how well do state-of-the-art GNNs actually perform under the challenging conditions of multi-label graphs?

Empirical evaluation revealed consistent shortcomings of existing methods, particularly their difficulty in disentangling overlapping label signals. This motivated the design of GNN-MultiFix, which integrates feature propagation with label propagation and positional encoding to address the specific challenges of MLNC. While effective in static graphs, this line of inquiry naturally extends to dynamic settings: if these challenges persist in static multi-label graphs, how are they amplified when the graph itself evolves over time?

This consideration leads to the setting of Continual Graph Learning (CGL), where both graph structure and label space evolve dynamically. Beyond the difficulties of noisy data and low homophily, models in this setting must contend with catastrophic forgetting as new tasks arrive. To systematically study these challenges, AGALE was proposed as a graph-aware continual learning evaluation framework, providing a principled way to benchmark models under evolving graph conditions.

Finally, the studies of static and continual learning setting highlight a broader issue: how should graph models be evaluated more comprehensively beyond narrow benchmarks and metrics? This question motivates a data-centric perspective on graph machine learning, emphasizing the role of instance-level graph properties in shaping performance and calling for evaluation frameworks that not only measure accuracy but also illuminate the interplay between data characteristics, and reliability of the model prediction.

1

INTRODUCTION

1.1. BACKGROUND

A **graph** is a fundamental data structure for modeling relationships between entities. Formally, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of *vertices* (or nodes) $\mathcal{V}$, which represent the entities, and a set of *edges* $\mathcal{E}$, which capture the connections between them. Depending on the application, edges may be *directed* or *undirected*, and they can also carry *weights* to encode additional properties such as cost, distance, or capacity. Graphs thus provide a flexible and expressive framework for representing complex systems with rich relational structure. They arise naturally in a wide range of domains, including social networks, biology, and knowledge representation. For example, in a protein–protein interaction network, proteins are modeled as nodes and their interactions as edges. In a social network, users correspond to nodes, while friendships or connections correspond to edges.

In recent years, Graph Neural Networks (GNNs) have emerged as the dominant paradigm for learning from graph-structured data [1–4]. These models are designed to leverage the relational inductive bias of graphs by iteratively propagating and aggregating information across connected nodes. Through this message-passing process, each node representation captures both its intrinsic attributes and the structural context of its neighborhood. Variants such as Graph Convolutional Networks (GCN) [2], Graph Attention Networks (GAT) [3], and GRAPHSAGE [4] have achieved strong results in canonical tasks that include node classification, link prediction, and graph-level classification.

The ability of GNNs to jointly encode feature and structural information has made them highly effective in real-world applications where relationships between entities are fundamental. They have become essential in domains such as traffic forecasting [5, 6], social network analysis [7, 8], and knowledge graph reasoning [9, 10]. For example, in recommender systems, GNNs capture user–item interactions as graphs to produce embeddings that reflect user preferences and collaborative patterns [11, 12]. In drug discovery, they predict molecular properties by learning from both atom-level features and the chemical structure of compounds [13, 14].

Yet, despite these remarkable successes, much of this progress is demonstrated on commonly used academic datasets and does not always extend to more realistic or complex environments. Current benchmarks disproportionately emphasize narrow domains such as two-dimensional molecular graphs, while overlooking broader and more impactful—yet inherently more complex—areas like combinatorial optimization and relational databases [15]. Within these broader applications, tasks such as multi-label node classification remain particularly challenging: nodes may belong to multiple categories simultaneously, and the resulting overlap and dependency among labels often violate the structural and statistical regularities that GNNs rely on for effective learning.

Beyond the challenges in the static graphs, continual graph learning (CGL) introduces another layer of difficulty: as both the node set and graph topology evolve over time, GNNs must adapt to new structures while avoiding catastrophic forgetting of previously acquired knowledge [16–20]. These limitations highlight the need for GNN architectures that are not only robust to noisy and overlapping label spaces but also adaptive to dynamic graph structures and memory efficient in retaining knowledge across time, thereby ensuring generalizable performance in both static and evolving real-world environments.

Building on these architectural challenges, another crucial limitation lies in how we analyze and evaluate the rapidly growing number of GNN variants. Although graph machine

learning models have achieved remarkable success, their performance is far from uniform across individual nodes. Standard aggregate metrics such as accuracy, F1 score, or AUROC, while useful for high-level benchmarking, obscure important node-level variations in model behavior and fail to reveal why certain nodes are consistently more difficult to classify. These difficulties often arise from intrinsic graph properties, such as feature sparsity (nodes with insufficient or noisy attributes), neighborhood label uncertainty (class diversity within a local neighborhood), or higher-order structural ambiguity (inconsistent label patterns among distant neighbors) that directly interact with how GNNs propagate and aggregate information. Crucially, models that achieve comparable overall accuracy may still generalize in fundamentally different ways, misclassifying disjoint subsets of nodes or exhibiting different levels of prediction stability and confidence. Developing systematic approaches to characterize this interplay between graph properties and model behavior is therefore essential for diagnosing hidden failure scenarios and ensuring the trustworthiness of graph machine learning systems.

Building on this introduction, the following section discusses the concrete challenges and research questions that we tackle in this thesis.

1.2. CHALLENGES AND RESEARCH GAPS

The node classification problem aims to predict labels for nodes by leveraging both graph topology and a subset of labeled nodes. While existing research has achieved notable progress in the multi-class setting—where each node is restricted to a single label—the more realistic multi-label scenario remains substantially more challenging. In multi-label graphs, a node may belong to several categories simultaneously, as in social networks where users hold multiple interests. This complicates the notion of label homophily, which measures the similarity among connected nodes based on shared categorical labels. Instead of neighbors sharing identical labels, they may only partially overlap. Consequently, the neighborhood aggregation step in GNNs often mixes heterogeneous label signals, making it difficult to extract the information most relevant to a target node's labels. This raises the first research question:

RQ 1. *To what extent do current GNNs improve on multi-label node classification tasks with more complicated homophily semantics?*

Beyond assessing the current progress of GNNs on multi-label node classification, a further challenge lies in identifying how their performance can be systematically improved. Recent studies suggest that incorporating positional encodings and label propagation into message passing can enrich node representations and help disambiguate overlapping label spaces. However, it remains unclear whether these enhancements consistently generalize across diverse datasets and graph structures, particularly in multi-label scenarios. This gap motivates the second research question:

RQ 2. *Does integrating positional encoding and label propagation into message-passing GNNs consistently benefit multi-label prediction on graph-structured data?*

In addition to the static setting, many real-world graphs evolve over time, shifting the task toward continual graph learning (CGL), which aims to adapt models to dynamic

graph structures and attributes while retaining previously learned knowledge. For example, in dynamic social networks, both the local topology (friendship links) and node labels (user interests) may change as the graph evolves. This introduces unique difficulties for incremental learning, such as adapting to new labels, expanding label sets, and mitigating data leakage across temporal partitions. Most existing frameworks, however, have been designed for single-label settings and struggle to accommodate the complexity of multi-label evolution. This leads to the third research question:

RQ 3. *Under which scenarios do current graph continual learning evaluation frameworks fail, and how can they be generalized for both multi- and single-label scenarios?*

Finally, even when focusing on static benchmarks, a critical limitation remains in how GNNs are evaluated. Conventional metrics such as accuracy, F1 score, or AUROC provide only aggregate views of performance, masking fine-grained instance-level variability. In practice, certain nodes may be persistently difficult to classify due to sparse features, noisy labels, or ambiguous structural patterns, and models with similar global accuracy may behave very differently at the node level. Without systematic profiling, we lack tools to uncover such differences and to link them back to underlying graph properties. This motivates the fourth research question:

RQ 4. *How can we understand the correlation between model behavior and graph data properties through instance-level profiling?*

Taken together, the above-discussed challenges highlight key research gaps in multi-label prediction, graph message-passing systems design, continual graph learning, and graph machine learning model evaluation. Having established the key challenges and research questions, we now delineate the scope of this thesis and summarize the contributions made toward addressing these issues.

1.3. THESIS SCOPE AND CONTRIBUTION

This dissertation addresses the research questions in the previous section through a sequence of studies that investigate the limitations of existing GNN models, develop new datasets and benchmarks, propose novel graph generation model and evaluation frameworks, and design diagnostic tools for fine-grained analysis of model behavior. We explain below how we addressed the research questions in more detail.

First, in addressing ***RQ1***, we conduct a systematic investigation into the problem of multi-label node classification with GNNs. We analyze the characteristics of widely used graph datasets, uncovering critical limitations such as imbalanced label distributions, weak label-induced similarities, and a large proportion of nodes with missing labels. To bridge the gap, we construct a benchmark of multi-label graph datasets, including curated biological graphs and a synthetic graph generator with tunable properties, which enables rigorous evaluation of learning methods under varying structural and semantic conditions. Through large-scale experiments across multiple datasets and algorithms, we reveal that simple baselines such as DEEPWALK [21] can outperform sophisticated GNNs, underscoring the importance of carefully characterizing datasets when developing new methods.

Then, in further answering ***RQ2***, we build on the insights gained through our above study, introducing GNN-MULTIFIX, a simple yet effective framework that fully leverages

node features, labels, and graph structure to address the limitations of existing GNNs in multi-label classification. Through theoretical analysis and extensive empirical evaluation, we demonstrate that GNN-MULTIFIX achieves superior performance compared to highly expressive GNN variants, even those designed to surpass the limitations of the 1-Weisfeiler-Lehman test. This work underscores the importance of designing models that integrate graph structure, node features, and suitable positional encodings. Such information is necessary to produce expressive node embeddings, since standard message-passing GNNs, when relying solely on neighborhood feature aggregation, are limited by their inherent locality and cannot distinguish nodes with identical local neighborhoods but different global positions.

Next, in addressing ***RQ3***, we extend our focus to the dynamic setting of continual graph learning (CGL), where models must adapt to sequentially arriving tasks without catastrophic forgetting. We develop a generalized evaluation framework for continual graph learning that is applicable to both multi-class and multi-label classification tasks, and further adaptable to graph- and edge-level prediction problems. Within this framework, we propose new data-splitting strategies to generate CGL benchmarks, theoretically analyze the properties of resulting subgraphs that are considered crucial for success of GNNs, and perform extensive experiments across methods from continual learning, dynamic graph learning, and CGL. Our analysis reveals the strengths and weaknesses of existing approaches, identifies their constraints, and highlights key directions for designing more robust and flexible methods for lifelong learning over graphs.

Finally, in addressing ***RQ4***, we turn to a fine-grained analysis of graph learning systems, examining how specific node-level properties relate to the model's confidence in predicting those nodes. We propose NODEPRO, a novel node profiling framework that provides fine-grained insights into model behavior by characterizing nodes along both data-centric and model-centric dimensions. Rather than focusing on improving accuracy, NODEPRO uncovers where and why models fail by profiling nodes based on their structural, feature-based, and label-related properties, and aligning these with prediction consistency and confidence. Through this approach, we demonstrate how NODEPRO not only explains node-level performance but also enables practical capabilities such as detecting misclassifications, identifying anomalous nodes, and diagnosing weaknesses in datasets or models that traditional evaluation metrics overlook.

With the scope and key contributions established, we now present an overview of the thesis structure to guide the reader through the subsequent chapters.

1.4. HOW TO READ THIS THESIS

- **Chapter 1: Introduction** – This chapter provides the foundational background necessary to follow the dissertation. Specifically, it introduces the node classification task on graph structured data, continual graph learning, and the emerging perspective of data-centric graph machine learning.

- **Chapter 2: Multi-label Node Classification on Graph-Structured Data** presents the study on the static multi-label node classification task in answering ***RQ1***. We conduct a detailed analysis of existing multi-label graph datasets, propose a graph generator with tunable parameters for controlled experimentation, and benchmark a range of models tailored to the multi-label setting.

- **Chapter 3: GNN-MultiFix – Addressing the Pitfalls of GNNs for Multi-label Node Classification** – In this chapter, we present a method designed for multi-label node classification in addressing ***RQ2***. The approach integrates feature propagation, label propagation, and positional encoding to generate informative node embeddings for the node classification task. We provide both theoretical and extensive empirical analysis to demonstrate its effectiveness and to compare against strong baseline methods.
- **Chapter 4: AGALE – A Graph-Aware Continual Learning Evaluation Framework** – This chapter extends continual graph learning to the multi-label setting in further answering of ***RQ3***. We identify the limitations of existing evaluation protocols, introduce new incremental learning settings, and design data-partitioning algorithms. Large-scale experiments are conducted to validate the effectiveness of the proposed framework.
- **Chapter 5: NodePro – An Instance-Level Profiling Framework for Graph-Structured Data** – In this chapter, we introduce a framework for profiling graph data at the instance level in answering ***RQ4***. By characterizing both structural and feature-based properties of nodes, the framework provides fine-grained insights into data characteristics and model behavior, offering a new perspective on graph machine learning model evaluation.
- **Chapter 6: Conclusion** – The final chapter summarizes the key contributions presented in Chapters 2 through 5 and outlines promising directions for future research in the context of multi-label graph learning, continual graph evaluation, and data-centric graph analysis.

1.5. THESIS ORIGINS

The four principal chapters (chapter 2 to 5) of this thesis originate from five individual publications completed during my Ph.D. studies. These works are the result of collaborations between myself, as the primary contributor, and my co-authors. The original papers, their publication venues, and their corresponding thesis chapters are listed below.

1. ***Multi-label Node Classification On Graph-Structured Data***
   Tianqi Zhao, Ngan Thi Dong, Alan Hanjalic, Megha Khosla. *Transactions on Machine Learning Research (TMLR)*, 2023, *Selected for Learning on Graphs Conference (LoG) 2024*, https://openreview.net/forum?id=EZhkV2BjDP.
   Linked to Chapter 2: Multi-label Node Classification on Graph-Structured Data

2. ***A data-centric approach for assessing progress of Graph Neural Networks***[1]
   Tianqi Zhao, Ngan Thi Dong, Alan Hanjalic, Megha Khosla. *Data-centric Machine Learning Research (DMLR) Workshop @ International Conference of Machine Learning (ICML)*, 2024, https://arxiv.org/pdf/2406.12439.
   Linked to Chapter 2: Multi-label Node Classification on Graph-Structured Data

3. ***GNN-MultiFix: Addressing the pitfalls for GNNs for multi-label node classification***
   Tianqi Zhao, Megha Khosla. *Under submission at Journal of Data-centric Machine Learning*

[1] This paper is a workshop paper corresponding to the first publication *Multi-label Node Classification On Graph-Structured Data.*

Research, https://arxiv.org/abs/2411.14094.
Linked to Chapter 3: GNN-MultiFix – Addressing the Pitfalls of GNNs for Multi-label Node Classification

4. ***AGALE: A Graph-Aware Continual Learning Evaluation Framework***
Tianqi Zhao, Alan Hanjalic, Megha Khosla. *Transactions on Machine Learning Research (TMLR)*, 2024, *Selected for conference presentation at Learning on Graphs Conference (LoG) 2024*, https://openreview.net/forum?id=xDTKRLyaNN.
Linked to Chapter 4: AGALE – A Graph-Aware Continual Learning Evaluation Framework

5. ***NodePro: An Instance-Level Profiling Framework for Graph-Structured Data***
Tianqi Zhao, Russa Biswas, Megha Khosla. *Under submission at Journal of Data-centric Machine Learning Research*, https://openreview.net/forum?id=CFzqKlPqMI.
Linked to Chapter 5: Instance-Level Profiling Framework for Graph-Structured Data

REFERENCES

1. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks* **20,** 61–80 (2009).
2. Kipf, T. N. & Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. arXiv: 1609.02907. http://arxiv.org/abs/1609.02907 (2016).
3. Veličković, P. *et al. Graph Attention Networks* 2018. arXiv: 1710.10903 [stat.ML]. https://arxiv.org/abs/1710.10903.
4. Hamilton, W. L., Ying, R. & Leskovec, J. *Inductive Representation Learning on Large Graphs* 2018. arXiv: 1706.02216 [cs.SI]. https://arxiv.org/abs/1706.02216.
5. Yu, B., Yin, H. & Zhu, Z. *Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting* in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)* (2018), 3634–3640.
6. Li, Y., Yu, R., Shahabi, C. & Liu, Y. *Diffusion convolutional recurrent neural network: Data-driven traffic forecasting* in *International Conference on Learning Representations (ICLR)* (2018).
7. Fan, W. *et al. Graph neural networks for social recommendation* in *The World Wide Web Conference (WWW)* (2019), 417–426.
8. Wu, Z. *et al.* A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* **32,** 4–24 (2020).
9. Schlichtkrull, M. *et al. Modeling relational data with graph convolutional networks* in *European Semantic Web Conference (ESWC)* (Springer, 2018), 593–607.
10. Wang, Q., Mao, Z., Wang, B. & Guo, L. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* **33,** 1549–1568 (2021).
11. Ying, R. *et al. Graph convolutional neural networks for web-scale recommender systems* in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)* (2018), 974–983.

12. Wang, X., He, X., Wang, M., Feng, F. & Chua, T.-S. *Neural graph collaborative filtering* in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)* (2019), 165–174.

13. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. *Neural message passing for quantum chemistry* in *Proceedings of the 34th International Conference on Machine Learning (ICML)* (2017), 1263–1272.

14. Duvenaud, D. K. *et al. Convolutional networks on graphs for learning molecular fingerprints* in *Advances in Neural Information Processing Systems (NeurIPS)* (2015), 2224–2232.

15. Bechler-Speicher, M. *et al. Position: Graph Learning Will Lose Relevance Due To Poor Benchmarks* in *Forty-second International Conference on Machine Learning Position Paper Track* (2025). https://openreview.net/forum?id=nDFpl2lhoH.

16. Wang, C., Qiu, Y., Gao, D. & Scherer, S. *Lifelong Graph Learning* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022), 13719–13728.

17. Wang, L., Zhang, X., Su, H. & Zhu, J. *A Comprehensive Survey of Continual Learning: Theory, Method and Application* 2023. arXiv: 2302.00487 [cs.LG].

18. Zhang, X., Song, D. & Tao, D. *CGLB: Benchmark Tasks for Continual Graph Learning* in *Advances in Neural Information Processing Systems* (eds Koyejo, S. *et al.*) **35** (Curran Associates, Inc., 2022), 13006–13021. https://proceedings.neurips.cc/paper_files/paper/2022/file/548a41b9cac6f50dccf7e63e9e1b1b9b-Paper-Datasets_and_Benchmarks.pdf.

19. Ko, J., Kang, S. & Shin, K. BeGin: Extensive Benchmark Scenarios and An Easy-to-use Framework for Graph Continual Learning. *arXiv preprint arXiv:2211.14568* (2022).

20. Zhao, T., Hanjalic, A. & Khosla, M. AGALE: A Graph-Aware Continual Learning Evaluation Framework. *Transactions on Machine Learning Research.* ISSN: 2835-8856. https://openreview.net/forum?id=xDTKRLyaNN (2024).

21. Perozzi, B., Al-Rfou, R. & Skiena, S. *Deepwalk: Online learning of social representations* in *Proc. of the International Conference on Knowledge Discovery and Data Mining* (2014).

2

Multi-label Node Classification Task On Graph-structured Data

[1]This chapter is based on the publication: **Tianqi Zhao**, Ngan Thi Dong, Alan Hanjalic, Megha Khosla, *Multi-label Node Classification On Graph-Structured Data*, *Transactions on Machine Learning Research (TMLR), Learning on Graphs Conference (LoG), 2023.*

[2]This chapter also builds on the workshop paper: **Tianqi Zhao**, Ngan Thi Dong, Alan Hanjalic, Megha Khosla, *A Data-Centric Approach for Assessing Progress of Graph Neural Networks*, *Data-centric Machine Learning Research (DMLR) Workshop at ICML, 2024.*

2.1. INTRODUCTION

Most of the existing works on graph node classification deploying Graph Neural Networks (GNNs) focus on a multi-class classification scenario while ignoring a more general and realistic scenario of multi-label classification, in which each node could have multiple labels. This scenario holds, for example in protein-protein interaction networks, in which each protein is labeled with multiple protein functions or associated with different diseases, or in social networks, where each user may carry multiple interest labels. In this work, we focus on multi-label node classification on graph-structured data with graph neural networks. For the sake of brevity, in what follows we will refer to multi-label node classification on graphs simply as multi-label node classification.

Regarding the approaches to deploying GNNs in a multi-label node classification scenario, a common practice is to transform the classification problem into multiple binary classification problems, one per label [1]. In other words, $|L|$ binary classifiers are trained, where each classifier j is responsible for predicting the $0/1$ association for the corresponding label $\ell_j \in L$. An assumption here, however, is that given the learned feature representations of the nodes, the labels are conditionally independent [2]. The validity of this assumption cannot be assured in a GNN-based learning approach as GNNs ignore the label correlation among the neighboring nodes and only focus on node feature aggregation during the representation learning step [2, 3].

Furthermore, we note that the success of GNNs is widely attributed to feature smoothing over neighborhoods and high label similarity among the neighboring nodes. Graphs with high similarity among labels of neighboring nodes are referred to as having high homophily. Alternatively, in heterophilic graphs, the labels of neighboring nodes usually disagree. Consequently, approaches like H2GCN [4] have been proposed, which claim a high performance on both homophilic and heterophilic node classification datasets. A subtle point here, however, is that a network with nodes characterized by multiple labels does not obey the crisp separation of homophilic and heterophilic characteristics. As an illustration, consider a friendship network with node labels representing user interests. Each user might share only a very small fraction of the interests with his friends, which indicates low homophily in the local neighborhood. Yet her/his interests/labels could be fully determined by looking at his one-hop neighbors. Therefore, the network is also not heterophilic in the sense that the connected users have similar interests. Consequently, the solutions taking into account higher-order neighborhoods to tackle low homophily might not always perform well in multi-label networks.

The lack of focused studies on multi-label node classification can also be attributed to the scarcity of available benchmark multi-label graph datasets. As an example, there is a single multi-label classification dataset, namely OGB-PROTEINS in the Open Graph Benchmark (OGB) [5]. More so, the OGB-PROTEINS dataset has around 90% of the nodes unlabeled in the test set. While the OGB leaderboard reflects benchmarking of a large number of methods on OGB-PROTEINS, the lack of labels in the test set combined with the use of the Area Under the ROC Curve (AUROC) metric leads to overly exaggerated performance scores. In particular, the performance is measured by the average of AUROC scores across the L (L is the total number of labels) binary classification tasks. In such a scenario, the model already achieves a very high score if it outputs a high probability for the negative class for each binary classification task.

Our Contributions. Our work thoroughly investigates the problem of multi-label node classification with GNNs. **Firstly**, we analyze various characteristics of multi-label graph-structured datasets, including label distribution, label induced first- and second-order similarities, that influence the performance of the prediction models. We observe that a large number of nodes in the current datasets only have a single label even if the average number of labels per node is relatively high. Moreover, for the popular OGB-PROTEINS dataset, around 89.38% of the nodes in the test set and 29.12% of train nodes have no label assigned.

Secondly, to remedy the gap of lack of datasets we build a benchmark of multi-label graph-structured datasets with varying structural properties and label-induced similarities. In particular, we curate 3 biological graph datasets using publicly available data. Besides, we develop a synthetic multi-label graph generator with tunable properties. The possibility to tune certain characteristics allows us to compare various learning methods rigorously.

Finally, we perform a large-scale experimental study evaluating 8 methods from various categories for the node classification task over 9 datasets. We observe that simple baselines like DEEPWALK outperform more sophisticated GNNs for several datasets. We present a comprehensive analysis of the performance of different methods based on their own and the dataset's characteristics.

2.2. RELATED WORK

Multi-label classification, which assigns multiple labels for each instance simultaneously, finds applications in multiple domains ranging from text classification to protein function prediction. In this work we focus on the case when the input data is graph-structured, for example, a protein-protein interaction network or a social network. For completeness, we also discuss the related works on using graph neural networks for multi-label classification on non-graph-structured data in Section 2.2.2. Several other paradigms of multi-label classification including extreme multi-label classification [6–8], partial multi-label classification [9, 10], multi-label classification with weak supervision [11, 12] (for a complete overview see the recent survey [13]). are out of the scope of the current work.

2.2.1. MULTI-LABEL CLASSIFICATION ON GRAPH-STRUCTURED DATA

Recent methods designed for multi-label node classification over graph-structured data can be categorized into four groups utilizing (1) node embedding approaches, (2) convolutional neural networks, (3) graph neural networks, and (4) the combination of label propagation and graph neural networks.

NODE REPRESENTATION OR EMBEDDING APPROACHES

[14–16] usually generate a lookup table of representations such that similar nodes are embedded closer. The learned representations are used as input features for various downstream prediction modules. While different notions of similarities are explored by different approaches, a prominent class of method is random walk based which defines similarity among nodes by their co-occurrence frequency in random walks. In this work, we specifically use DEEPWALK [14] as a simple baseline that uses uniform random walks to define node similarity.

Other methods like [17–19] use **convolutional neural networks** to first extract node representations by aggregating feature information from its local neighborhood. The extracted

feature vectors are then fused with label embeddings to generate final node embeddings. Finally, these node embeddings are used as input for the classification model to generate node labels. In this work, we adopt LANC [18] as a baseline from this category, as previous works [18, 19] have shown its superior performance compared to other commonly used baselines for the multi-label node classification task.

GRAPH NEURAL NETWORKS(GNNS)

popularised by graph convolution network [20] and its variants compute node representation by recursive aggregation and transformation of feature representations of its neighbors which are then passed to a classification module. Let $\mathbf{x}_i^{(k)}$ be the feature representation of node i at layer k, $\mathcal{N}(i)$ denote the set of its 1-hop neighbors. The $k-th$ layer of a graph convolutional operation can then be described as

$$\mathbf{z}_i^{(k)} = \text{AGGREGATE}\left(\left\{\mathbf{x}_i^{(k-1)}, \left\{\mathbf{x}_j^{(k-1)} \mid j \in \mathcal{N}(i)\right\}\right\}\right), \quad \mathbf{x}_i^{(k)} = \text{TRANSFORM}\left(\mathbf{z}_i^{(k)}\right)$$

For the multi-label node classification, a sigmoid layer is employed as the last layer to predict the class probabilities: $\boldsymbol{y} \leftarrow (\text{sigmoid}(\boldsymbol{z}_i^{(L)}\theta))$, where θ corresponds to the learnable weight matrix in the classification module. GNN models mainly differ in the implementation of the aggregation layer. The simplest model is the graph convolution network (GCN) [20] which employs degree-weighted aggregation over neighborhood features. GAT [21] employs several stacked Graph Attention Layers, which allows nodes to attend over their neighborhoods' features. GRAPHSAGE [22] follows a sample and aggregate approach in which only a random sample of the neighborhood is used for the feature aggregation step. GNNs, in general, show better performance on high homophilic graphs in which the connected nodes tend to share the same labels. Recent approaches like H2GCN [4] show improvement on heterophilic graphs (in the multi-class setting). Specifically, it separates the information aggregated from the neighborhood from that of the ego node. Further, it utilizes higher-order neighborhood information to learn informative node representations.

LABEL PROPAGATION WITH GNNS

Prior to the advent of GNNs label propagation (LPA) algorithms constituted popular approaches for the task of node classification. Both LPA and GNNs are based on message passing. While GNNs propagate and transform node features, LPA propagates node label information along the edges of the graph to predict the label distribution of the unlabelled nodes. A few recent works [23, 24] have explored the possibilities of combining LPA and GNNs. [23] employs knowledge distillation in which the trained GNN model (teacher model) is distilled into a combination of GNN and parameterized label propagation module. GCN-LPA [24] utilizes LPA serves as regularization to assist the GCN in learning proper edge weights that lead to improved classification performance. Different from our work both of the above works implicitly assume a multi-class setting. Moreover, [23] focuses on the interpretable extraction of knowledge from trained GNN and can only discover the patterns learned by the teacher GNN.

2.2.2. MULTI-LABEL CLASSIFICATION ON NON-GRAPH-STRUCTURED DATA USING GNNS

There has also been an increasing trend to use graph neural networks to exploit the implicit relations between the data and labels in multi-label classification problems on non-graph data. For example, [25] models the problem of extreme classification as that of link prediction in a document-label bipartite graph and uses graph neural networks together with the attention mechanism to learn superior node representations by performing graph convolutions over neighborhoods of various orders. ML-GCN[17] generates representations for the images using CNN and extracts label correlation by constructing a label-label graph from the label co-occurrence matrix. LaMP[26] treats labels as nodes on a label-interaction graph and computes the hidden representation of each label node conditioned on the input using attention-based neural message passing. Likewise, for the task of multi-label image recognition [27] builds a directed graph over the object labels, where each label node is represented by word embeddings of the label, and GCN is employed to map this label graph into a set of inter-dependent object classifiers. The above works and many more like [28–35] are not part of the current study as they are developed for non-graph structured data.

2.3. A DETAILED ANALYSIS OF EXISTING AND NEW DATASETS

We commence by analyzing various properties of existing multi-label datasets including label distributions, label similarities, and cross-class neighborhood similarity(CCNS), which could affect the performance of prediction models. In section 2.3.2 we further curate new real-world biological datasets which to some extent improve the representativeness of multi-label graph datasets. Finally, in Section 2.4 we propose our synthetic multi-label graph generator to generate multi-label graph datasets with tunable properties. The possibility to control specific influencing properties of the dataset allows us to benchmark various learning methods effectively. We will need the following quantification of label similarity in multi-label datasets.

Label homophily. The performance of GNNs is usually argued in terms of label homophily which quantifies similarity among the neighboring nodes in the graph. In particular, label homophily is defined in [4] as the fraction of the homophilic edges in the graph, where an edge is considered homophilic, if it connects two nodes with the same label. This definition can not be directly used in the multi-label graph datasets, as each node can have more than one label and it is rare in the multi-label datasets that the whole label sets of two connected nodes are the same. Usually, two nodes share a part of their labels. We, therefore, propose a new metric to measure the label homophily h of the multi-label graph datasets as follows.

Definition 1. *Given a multi-label graph $\mathcal{G}$, the label homophily h of $\mathcal{G}$ is defined as the average of the Jaccard similarity of the label set of all connected nodes in the graph:*

$$h = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \frac{|\ell(i) \cap \ell(j)|}{|\ell(i) \cup \ell(j)|}.$$

Label homophily is a first-order label-induced similarity in that it quantifies the similarity among neighboring nodes based on their label distributions.

Cross-Class Neighborhood Similarity for Multi-label graphs. Going beyond the label similarity among neighboring nodes, we consider a second order label induced metric which quantifies the similarity among neighborhoods of any two nodes. [36] introduced Cross-Class Neighborhood Similarity (CCNS) for multi-class graphs. Using CCNS, [36] attributes the improved performance of GNNs to the higher similarity among neighborhood label distributions of nodes of the same class as compared to different classes. We extend their proposed CCNS measure to analyze multi-label datasets. Given two classes c and c', the CCNS score measures the similarity in the label distributions of neighborhoods of nodes belonging to c and c' and is defined as follows. One can visualize the CCNS scores between all class pairs in an $C \times C$ matrix where C is the total number of classes. Common GNNs are expected to perform better on datasets which higher scores on the diagonal than off-diagonal elements of the corresponding CCNS matrix.

Definition 2. *Given a multi-label graph $\mathcal{G}$ and the set of node labels Y for all nodes, we define the multi-label cross-class neighborhood similarity between classes $c, c' \in C$ is given by*

$$s(c, c') = \frac{1}{|\mathcal{V}_c \| \mathcal{V}_{c'}|} \sum_{i \in \mathcal{V}_c, j \in \mathcal{V}_{c'}, i \neq j} \frac{1}{|\ell(i)||\ell(j)|} \cos(\mathbf{d}_i, \mathbf{d}_j), \tag{2.1}$$

where $\mathcal{V}_c = \{i | c \in \ell(i)\}$ is the set of nodes with one of their labels as c. The vector $\mathbf{d}_i \in \mathbb{R}^C$ corresponds to the empirical histogram (over $|C|$ classes) of node i's neighbors' labels, i.e., the c^{th} entry of $\mathbf{d}_i$ corresponds to the number of nodes in $\mathcal{N}(i)$ that has one of their label as c and the function $cos(.,.)$ measures the cosine similarity and is defined as

$$cos(\mathbf{d}_i, \mathbf{d}_j) = \frac{\mathbf{d}_i \cdot \mathbf{d}_i}{||\mathbf{d}_i|| ||\mathbf{d}_j||}.$$

Note that as a node in a multi-label dataset could belong to multiple classes we exclude the possibility of comparing a node to itself. By introducing a factor of $|\ell(i)||\ell(j)|$ in the denominator we are able to normalize the contribution of multi-labeled nodes for several class pairs.

2.3.1. EXISTING DATASETS

We start by analyzing the four popular multi-label node classification datasets: (i) BLOGCAT [37], in which nodes represent bloggers and edges their relationships, the labels denote the social groups a blogger is a part of, (ii)YELP [38], in which nodes correspond to the customer reviews and edges to their friendships with node labels representing the types of businesses and (iii) OGB-PROTEINS [5], in which nodes represent proteins, and edges indicate different types of biologically meaningful associations between the proteins, such as physical interactions, co-expression, or homology [39, 40]. The labels correspond to protein functions. (iiii) DBLP [41], in which nodes represent authors and edges the co-authorship between the authors, and the labels indicate the research areas of the authors.

We report in Table 2.1 the characteristics of these datasets including the label homophily. In the following, we discuss in detail the various characteristics of these datasets along with their limitations for the effective evaluation of multi-label classification.

Skewed label distributions. Figure 2.1 illustrates the label distributions in the four datasets. Quantitatively 72.34% nodes in BLOGCAT only have one label. However, the most labeled

Table 2.1: Dataset statistics. $|\mathcal{V}|$ and $|\mathcal{E}|$ denote the number of nodes and edges in the graph. $|\mathcal{F}|$ is the dimension of the node features. $clus$ and r_{homo} denote the clustering coefficient and the label homophily. C indicates the size of all labels in the graph. ℓ_{med}, ℓ_{mean}, and ℓ_{max} specify the median, mean, and max values corresponding to the number of labels of a node. '25%', '50%', and '75%' corresponds to the 25th, 50th, and 75th percentiles of the sorted list of the number of labels for a node. "N.A." means the corresponding characteristic is not available in the graph.

DATASET	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{F}\|$	$clus$	r_{homo}	C	ℓ_{med}	ℓ_{mean}	ℓ_{max}	25%	50%	75%
BLOGCAT	10K	333K	N.A.	0.46	0.10	39	1	1.40	11	1	1	2
YELP	716K	7.34M	300	0.09	0.22	100	6	9.44	97	3	6	11
OGB-PROTEINS	132K	39M	8	0.28	0.15	112	5	12.75	100	0	5	20
DBLP	28K	68K	300	0.61	0.76	4	1	1.18	4	1	1	1

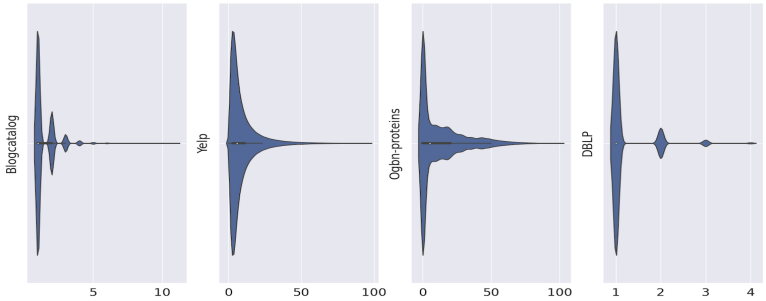
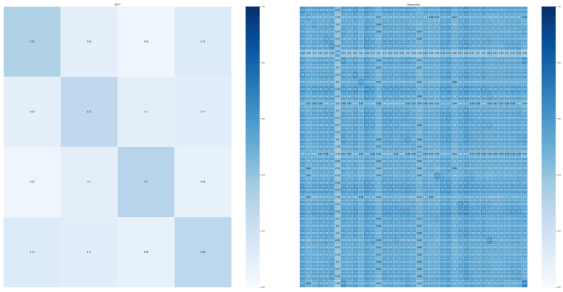


Figure 2.1: Label distributions. In BLOGCAT, the majority of the nodes have one label. In OGB-PROTEINS, around 41% of total nodes have no labels, and only three nodes have the maximum number of 100 labels.

data points are assigned with 11 labels. YELP has a total of 100 labels, the most labeled data points have 97 labels, whereas over 50% of the nodes have equal or less than 5 labels. Nevertheless, YELP exhibits a high multi-label character with 75% of the nodes with more than 3 labels. OGB-PROTEINS is an extreme case in which 40.27% of the nodes do not have any label. DBLP is the dataset with the highest portion of nodes with single labels, with the exact percentage of 85.4%.

Issue in evaluation using AUROC scores under high label sparsity. Another so far unreported issue in multi-label datasets is the unlabeled data. In OGB-PROTEINS, 40.27% of nodes do not have labels. Moreover, 89.4% of the test nodes are unlabelled. More worrying is the use of the AUROC score metric in the OGB leaderboard to benchmark methods for multi-label classification. In particular, a model that assigns "No Label" to each node (i.e. predict negative class corresponding to each of the independent L binary classification tasks) will already show a high AUROC score. We in fact observed that increasing the number of training epochs (which encourage the model to decrease training loss by predicting the negative class) increased the AUROC score whereas other metrics like AP or F1 score dropped or stayed unchanged.

Cross-class neighborhood similarity. In Figure 2.2 we visualize the cross-class neighborhood similarity matrix for DBLP and BLOGCAT. The cells on the diagonal reflect the



(a) Cross class Neighborhood Similarity in DBLP

(b) Cross class Neighborhood Similarity in BLOGCAT

Figure 2.2: Cross class Neighborhood Similarity in real-world datasets

intra-class neighborhood similarity, whereas the other cells indicate the inter-class neighborhood similarity computed using (2.1). The contrast in 2.2a means that nodes from the same class tend to have similar label distribution in their neighborhood, while nodes from different classes have rather different label distributions in their neighborhoods. We will later see in the experimental section that GNNs indeed benefit from this characteristic to identify correctly the nodes in the same classes in DBLP. On the contrary, the intra- and inter-class similarity are more similar in BLOGCAT, making it intricate for GNNs to classify the nodes to their corresponding classes.

2.3.2. NEW BIOLOGICAL INTERACTION DATASETS

Motivated by the natural applicability of the multi-label classification task in various biological datasets and to improve the representativeness of available datasets, we collect three real-world biological datasets corresponding to different multi-label classification problems: the PCG dataset for the protein phenotype prediction, the HUMLOC, and EUKLOC datasets for the human and eukaryote protein subcellular location prediction tasks, respectively. On each dataset, we build a graph in which each protein is modeled as a node. The node label is the corresponding protein's label. An edge represents a known interaction between two proteins retrieved from a public database. The detailed pre-processing steps and the original data sources are discussed in Appendix 2.8.1, 2.8.1, and 2.8.1. Table 2.2 presents an overview of the three datasets' characteristics.

Table 2.2: Statistics for new datasets. The column notations are the same as in Table 2.1.

DATASET	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{F}\|$	$clus$	r_{homo}	C	ℓ_{med}	ℓ_{mean}	ℓ_{max}	25%	50%	75%
PCG	3K	37k	32	0.34	0.17	15	1	1.93	12	1	1	2
HUMLOC	3.10k	18K	32	0.13	0.42	14	1	1.19	4	1	1	1
EUKLOC	7.70K	13K	32	0.14	0.46	22	1	1.15	4	1	1	1

While all the existing datasets had very low homophily, HUMLOC, and EUKLOC show higher homophily. Moreover, these datasets improve the representativeness in terms of varying graph structure (reflected in computed clustering coefficient) and in node, edge, and

feature sizes. On the downside, these datasets also show a similar low multi-label character, with the majority of nodes in these datasets still having a single label.

Among the three datasets, PCG shows a bit more balanced label distribution (see Figure 2.3) as compared to the other two. Figure 2.4 provides the cross-class neighborhood similarity scores. All three datasets show different patterns according to CCNS measure which is desirable to analyse the differences in method's performance. While in PCG we see an overall high scores for CCNS, the difference in inter- and intra- class similarities is not prominent. HUMLOC shows a slightly more contrasting intra- and inter-class neighborhood similarity. EUKLOC, on the other hand, show very small neighborhood label similarities for nodes of same or different classes.

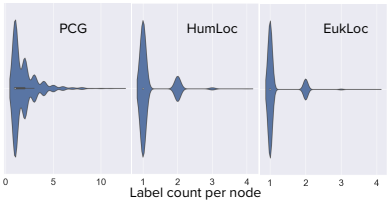
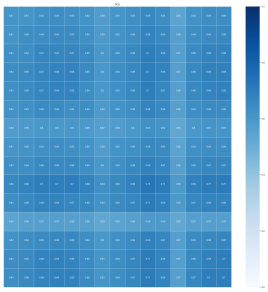
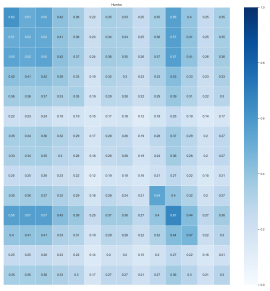


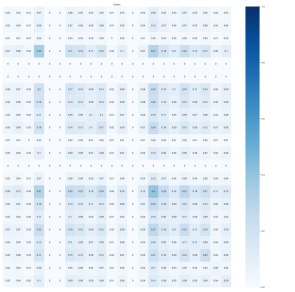
Figure 2.3: Label distributions in biological datasets. The majority of the nodes in all datasets have one label.



(a) Cross class Neighborhood Similarity in PCG



(b) Cross class Neighborhood Similarity in HUMLOC



(c) Cross class Neighborhood Similarity in EUKLOC

Figure 2.4: Cross class Neighborhood Similarity in real-world datasets and proposed biological datasets

2.4. MULTI-LABEL GRAPH GENERATOR FRAMEWORK

In the previous sections, we analyzed various real-world dataset properties which could influence a method's performance. We now develop a multi-label graph generator that will allow us to build datasets with tunable properties for a holistic evaluation. With our proposed framework we can build datasets with *high multi-label character, varying feature quality, varying label homophily, and CCNS similarity*. We now describe the two main steps of our multi-label graph generator.

Multi-label generator. In the first step, we generate a multi-label dataset using MLDATAGEN [42]. We start by fixing the total number of labels and features. We then construct a hypersphere, $H \in \mathbb{R}^{|\mathscr{F}|}$ centered at the origin and has a unit radius. Corresponding to each label in set $\mathscr{L}$ we then generate a smaller hypersphere with a random radius but with the condition that it is contained in H. We now start populating the smaller hyperspheres with randomly generated datapoints with $|\mathscr{F}|$ dimensions. Note that each datapoint may lie in

a number of overlapping hyperspheres. The labels of the datapoint then correspond to the hyperspheres it lies in.

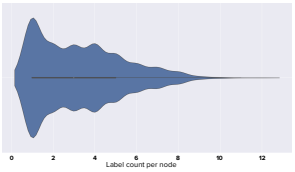
Graph generator. Having constructed the multi-label dataset, we now construct edges between the data points by using a social distance attachment model [43]. In particular, for two given datapoints (nodes) i and j, their corresponding feature vectors are their coordinates given $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively. The corresponding label vectors are denoted by $\mathbf{y}_i$ and $\mathbf{y}_j$. We denote the hamming distance between the label vectors of nodes i and j by $d(\mathbf{y}_i, \mathbf{y}_j)$. We then construct an edge between datapoints (nodes) i and j, (i, j) with probability given by

$$p_{ij} = \frac{1}{1 + [b^{-1} d(\mathbf{y}_i, \mathbf{y}_j)]^{\alpha}} \tag{2.2}$$

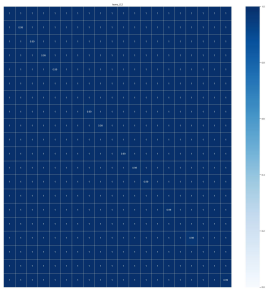
where α is a homophily parameter, b is the characteristic distance at which $p_{ij} = \frac{1}{2}$. Note that the edge density is dictated by both the parameters α and b. A larger b would result in denser graphs. A larger homophily parameter α would assign a higher connection probability to the node pairs with shorter distances or nodes with similar labels. In particular, it is a random geometric graph model, which in the limit of large system size (number of nodes) and high homophily (large α) leads to sparsity, non-trivial clustering coefficient and positive degree assortativity [44], properties exhibited by real-world networks. By using different combinations of values of α and b, we can control the connection probability and further the label homophily of the generated synthetic graphs. We perform an extensive empirical analysis to study the relationship between α, b, and the label homophily of the generated datasets. We provide detailed instructions to use our synthetic data generator and our empirical analysis in Appendix 2.8.2.

Synthetic datasets with fixed homophily and varying feature quality. For our experimental analysis, we generate synthetic datasets with $3K$ nodes, 10 features, and a total of 20 labels. Towards analyzing the variation in the method's performance with variation in *homophily* and *feature quality* first we constructed a dataset with fixed homophily of 0.377 (using $\alpha = 8.8$, $b = 0.12$) and edge set of size $1M$. We refer to this dataset as SYNTHETIC1. We create five variants of SYNTHETIC1 with varying feature quality. In particular, we add 10 random features for every node, which we refer to as *irrelevant* features. We then generate its variants by removing original features such that the ratio of the number of original to that of irrelevant features varies as in $\{1, 0.8, 0.5, 0.2, 0\}$. From the label distribution plot in 2.5a, we observe the dataset is more multi-label than the real-world datasets because a higher number of nodes now have multiple labels.

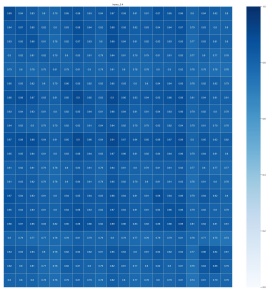
Synthetic datasets with varying homophily and CCNS. We also use 5 different pairs of α and b and the same multi-label data from the first step to construct 5 synthetic graphs with label homophily (rounded up) in $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ to conduct the experiment where we test the influence of the label homophily on the performances of the node classification methods. The detailed statistics of the synthetic datasets are provided in Appendix 2.8.2 in Table 2.5. Figure 2.5 visualizes the cross-class neighborhood similarity in the five hypersphere datasets with varying homophily levels (homophily varies in [0.2, 0.4, 0.6, 0.8, 1.0]). In the synthetic graphs with low label homophily, the intra- and inter-class neighborhood similarities show no significant differences, i.e., the nodes from different classes have similar label distributions in their neighborhoods. We overall observe a high absolute value of neighborhood similarity. The reason here is that similar to BLOGCAT



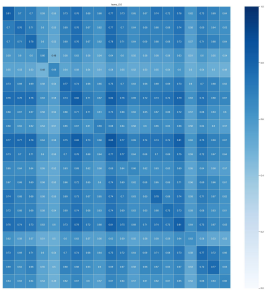
(a) Label distribution in the synthetic dataset is more balanced. A relatively high multi-label character is exhibited with 50% nodes having more than 3 labels.



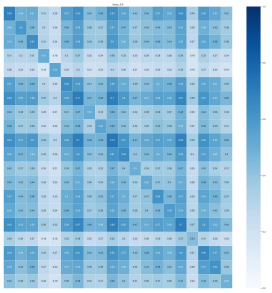
(b) Cross-class Neighborhood Similarity in graph label homophily=0.2



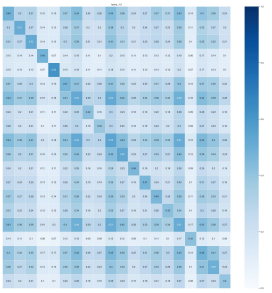
(c) Cross-class Neighborhood Similarity in graph label homophily=0.4



(d) Cross-class Neighborhood Similarity in graph label homophily=0.6



(e) Cross-class Neighborhood Similarity in graph label homophily=0.8



(f) Cross-class Neighborhood Similarity in graph label homophily=1.0

Figure 2.5: Cross-class Neighborhood Similarity in hypersphere datasets with varying label homophily

(which also shows overall high CCNS scores) these synthetic low homophily graphs are highly connected and have a high average degree. As the label homophily gets higher, the contrast between the intra- and inter-class similarity becomes more significant.

2.5. EXPERIMENTS

From our dataset analysis in the previous sections we observe that, unlike commonly used multi-class datasets, multi-label datasets usually have low label homophily and a more varied cross-class neighborhood similarity. Moreover, similar to the case of multi-class datasets, node features might not always be available (as in BLOGCAT) or might be noisy. In this section, we perform a large-scale empirical study comparing 8 methods over 7 real-world multi-label datasets and 2 sets of synthetic datasets with varying homophily and feature quality. Our experiments are designed to reveal and understand (i) properties of datasets that favor certain methods over others and (ii) the effect of varying feature quality and label homophily on method performance. The training and hyperparameter settings for each model are summarized in the Appendix 2.8.3 in tables 2.7 and 2.8. Our code is available at https://github.com/Tianqi-py/MLGNC.

Datasets. We employ 7 real-world datasets including BLOGCAT, YELP, OGB-PROTEINS,

DBLP, PCG, HUMLOC, EUKLOC, and 2 sets of synthetic datasets with varying homophily and feature quality. These datasets are already described in Section 2.3. For all datasets except OGB-PROTEINS, HUMLOC, and EUKLOC we generate 3 random training, validation, and test splits with 60%, 20%, and 20% of the data. For OGB-PROTEINS, HUMLOC, and EUKLOC we follow the predefined data splits from [5], [45] and [46] respectively. As BLOGCAT has no given node features, we use an identity matrix as the input feature matrix.

Compared Methods. For a holistic evaluation we include four classes of compared methods (i) simple methods, which include Multilayer Perceptron (MLP), which only uses node features and ignores the graph structure, and DEEPWALK, which only uses graph structure and ignores the node features (ii) Convolutional neural networks based which employ convolutional operations to extract representations from node's local neighborhoods and merge them with label embeddings for the final classification. We choose LANC as a baseline from this category, as previous works [18, 19] have shown its superior performance. (iii) graph neural networks including (a) GCN, GAT, and GRAPHSAGE, which are known to perform well for graphs with high label homophily, and (b)H2GCN, which is designed to perform well both on homophilic and heterophilic graphs and (iiii) GCN-LPA which combines label propagation and GCN for node classification. All these methods are also discussed in Section 3.2.

Evaluation Metrics. We report the average micro- and macro-F1 score, macro-averaged AUC-ROC score, macro-averaged average precision score, and standard deviation over the three random splits. Due to space constraints, we report average precision (AP) in the main paper, and all detailed results are available in Tables 2.9, 2.10, 2.11 in the Appendix 2.8.4. Our choice of using AP over AUROC as the metric is also motivated in Appendix 2.8.5.

2.6. RESULTS AND DISCUSSION

Table 2.3: Mean performance scores (Average Precision) on real-world datasets. The best score is marked in bold. The second best score is marked with underline. "OOM" denotes the "Out Of Memory" error.

Method	BLOGCAT	YELP	OGB-PROTEINS	DBLP	PCG	HUMLOC	EUKLOC
MLP	0.043	0.096	0.026	0.350	0.148	0.170	0.120
DEEPWALK	**0.190**	0.096	0.044	0.585	**0.229**	0.186	0.076
LANC	<u>0.050</u>	OOM	<u>0.045</u>	0.836	0.185	0.132	0.062
GCN	0.037	0.131	**0.054**	**0.893**	<u>0.210</u>	**0.252**	**0.152**
GAT	0.041	0.150	0.021	0.829	0.168	<u>0.238</u>	<u>0.136</u>
GRAPHSAGE	0.045	**0.251**	0.027	<u>0.868</u>	0.185	0.234	0.124
H2GCN	0.039	<u>0.226</u>	0.036	0.858	0.192	0.172	0.134
GCN-LPA	0.043	0.116	0.023	0.801	0.167	0.150	0.075

2.6.1. RESULTS ON REAL-WORLD DATASETS.

Table 2.3 we provide the results for 7 real-world datasets. In general, on datasets with low label homophily, such as BLOGCAT and PCG, node representation or embedding learning methods such as DEEPWALK, outperform more sophisticated GNN based methods and simple MLP baseline. Classical GNNs show better performance on datasets characterized by

high label homophily. H2GCN which is designed for multi-class datasets with heterophily do not show a performance improvement over classical GNNs on multi-label graph datasets with low homophily. Likewise, the method that combine label propagation with GNNs, achieve only comparable results to classical GNNs.

BLOGCAT. For BLOGCAT all GNN approaches as well as LANC and GCN-LPA obtain scores close to that of MLP which do not use any graph structure. Notably, MLP uses identity matrix as input features which in principle provides no useful information. The corresponding scores can be seen as the result of a random assignment. As a method unifying the label and feature propagation, GCN-LPA does not show improvement on BLOGCAT compared to other baselines. This is because GCN-LPA uses the label propagation to adjust the edge weight and still only generates embedding by aggregating features over the weighted graph, while DEEPWALK take advantage of the informative topological structure in the graph and achieves the best performance.

YELP, OGB-PROTEINS and DBLP. YELP has low label homophily and low clustering coefficient but a more balanced label distribution as compared to other real-world datasets, so approaches designed specifically for low homophilic graphs and are capable of preserving information from both low- and high- order neighborhoods are expected to perform better for this dataset. Among the GNN-based baselines, H2GCN which has shown improvements in low homophilic multi-class datasets outperforms GCN,GAT, and GCN-LPA but is still outperformed by GRAPHSAGE. The use of CNNs in LANC to aggregate neighborhood features leads to excessive memory utilization for a graph with a very high maximum degree. This led to the out-of-memory error for YELP.

All methods perform poorly on OGB-PROTEINS with GCN and LANC slighty outperforming others. It also has low homophily which is similar to BLOGCAT, which also does not provide GNNs with any additional advantage. In particular, DEEPWALK and H2GCN outperform GAT and GRAPHSAGE as well as more sophisticated GCN-LPA. It is worth noting that the previously reported results on the OGB-PROTEINS leaderboard [5] are significantly exaggerated due to the utilization of the AUROC metric in conjunction with excessive model training. When using the metic Average Precision, the scores we get is much lower that what were reported on the leaderboard.

As a co-authorship dataset, DBLP has the highest label homophily and the largest portion of nodes with a single label among all the real-world datasets. As shown in Figure 2.2a in the section 2.3.1, the inter-class similarity is much weaker than the intra-class similarity. Besides, the high clustering coefficient indicates that the local neighborhood is highly connected. All of these factors further justify the best performance shown by GCN. Besides, the relatively poor performances of MLP and DEEPWALK indicate that the features or the structure alone are not sufficient for estimating node labels.

PCG, HUMLOC and EUKLOC. If the features are highly predictive of the labels, the simple baselines MLP using only feature information would be a competitive baseline. In the experiments with the datasets, where the features are highly correlated (evident from relatively better performance of MLP) with the assigned labels, i.e., the biological interaction datasets HUMLOC and EUKLOC (shown in Table 2.3), GNNs and GCN-LPA tend to have better performance than DEEPWALK and LANC which only utilizes graph structure and features from the direct neighborhood. PCG exhibits low label homophily and high clustering

coefficient. Consistent with observations in other low homophilic datasets DEEPWALK outperforms other methods in PCG too. H2GCN, on the other hand, is outperformed by simpler GNN baseline like GCN.

2.6.2. RESULTS ON SYNTHETIC DATASETS

Table 2.4 provides results for synthetic datasets with varying feature quality and label homophily. We provide a detailed analysis and argue about performance differences in the following sections.

Table 2.4: Average Precision (mean) on the synthetic datasets with varying levels of feature quality and homophily parameter. r_{ori_feat} and r_{homo} refer to the fraction of original features and the homophily parameter value, respectively.

Method	r_{ori_feat}					r_{homo}				
	0.0	0.2	0.5	0.8	1.0	0.2	0.4	0.6	0.8	1.0
MLP	0.172	0.187	0.220	0.277	0.343	**0.343**	0.343	0.343	0.343	0.343
DEEPWALK	**0.487**	**0.487**	**0.487**	**0.487**	**0.487**	0.181	**0.522**	**0.813**	**0.869**	0.552
LANC	0.337	0.342	0.365	0.353	0.391	0.190	0.380	0.434	0.481	0.629
GCN	0.313	0.316	0.311	0.301	0.337	0.261	0.343	0.388	0.450	0.493
GAT	0.311	0.339	0.329	0.338	0.360	0.172	0.359	0.390	0.428	0.439
GRAPHSAGE	0.300	0.328	0.377	0.393	0.430	0.289	0.426	0.458	0.533	0.553
H2GCN	0.376	0.401	0.427	0.442	0.467	0.297	0.484	0.512	0.572	**0.652**
GCN-LPA	0.337	0.333	0.368	0.363	0.391	0.170	0.408	0.495	0.604	0.583

EFFECT OF VARYING FEATURE QUALITY.

As simple baseline MLP and other GNN-based models use features as input, we assume they will be sensitive to the varying feature quality. As DEEPWALK uses the graph structure alone, its performance would not be affected by the varying feature quality. To further validate our hypothesis and test the robustness of the methods to feature quality, we compare the method performances on variants of the generated SYNTHETIC1 dataset. Specifically, we vary the ratio of the original to the irrelevant features as in $\{0, 0.2, 0.5, 0.8, 1.0\}$.

As shown in Table 2.4, under all levels of feature quality, the performances of DEEPWALK do not change, as it generates representations for the nodes solely from the graph structure. The MLP is unsurprisingly the most sensitive method to the varying feature quality because it completely ignores the graph structure. LANC is also sensitive to the change of the feature quality as it extracts feature vectors from the local neighborhood by performing convolutional operations on the stacked feature matrix of the direct neighbors.

The SYNTHETIC1 dataset used in this experiment has a label homophily of 0.3768, which is a relatively high label homophily for the multi-label datasets. Surprisingly, GCN-LPA which employs the label information performs only a little better than GRAPHSAGE on the feature-to-noise ratio of 0 and 0.2. H2GCN, on the other hand, outperforms all GNN-based baselines and GCN-LPA for all levels of feature quality.

EFFECT OF VARYING LABEL HOMOPHILY.

In this subsection, we test the robustness of the methods to varying homophily and further argue how low label homophily has different semantics on multi-label graphs as it does on

multi-class datasets. Specifically, we vary the label homophily as in $\{0.2, 0.4, 0.6, 0.8, 1.0\}$.

As shown in Table 2.4, the performances of MLP do not change under different levels of label homophily, due to the fact that MLP only use the input features.

H2GCN is a method that has been shown to perform well on heterophilic multi-class datasets. In the multi-label scenarios, it exhibits better performance than other GNN methods but is outperformed by simple MLP baseline in case of label homophily of 0.2.

On the other hand, with most of the attention drawn to developing new complicated methods for the node classification task, we observe simple baselines such as DEEPWALK outperform standard GNNs in several scenarios. On the synthetic datasets with the label homophily of 0.4, 0.6 and 0.8, DEEPWALK is the best-performing method. As shown in Table 2.4, the performance of DEEPWALK drops when the label homophily is 1.0. However, we want to emphasize that this is because we fixed the same walk length for DEEPWALK for all levels of label homophily and the improvement can be shown with a possible hyperparameter tuning.

As mentioned in section 2.4, the intra-class similarity is significantly stronger than the inter-class similarity in the synthetic graphs with higher label homophily, which helps GNN-based models to better distinguish the nodes from different classes and thus achieve better results in the node classification task.

2.7. CONCLUSION

We investigate the problem of multi-label node classification on graph-structured data. Filling in the gaps in current literature, we (i) perform in-depth analysis on the commonly used benchmark datasets, create and release several real-world multi-label datasets and a graph generator model to produce synthetic datasets with tunable properties, (ii) compare and analyse the performances of the methods from different categories for the node classification task by conducting large-scale experiments on 9 datasets and 8 methods, and (iii) release our benchmark publicly.

We have novel and compelling insights from our analysis of specific datasets and GNN approaches. For instance, we uncover the pitfalls of the commonly used OGB-Protein dataset for model evaluation. While multi-label graph datasets usually show low homophily, we show that approaches working on low homophilic multi-class datasets cannot trivially work on multi-label datasets which usually have low homophily.

While current graph-based machine learning methods are usually evaluated on multi-class datasets, we demonstrate that the acquired improvements cannot always be translated to the more general scenario when the nodes are characterized by multiple labels. We believe that our work will open avenues for more future work and bring much-deserved attention to multi-label classification on graph-structured data. In future work, we plan to study the interplay of different dataset characteristics (for example edge density and label homophily) on the model performance.

2.8. APPENDIX

Organization. We explain the construction details of the biological datasets in 2.8.1. Furthermore, in Section 2.8.2, we study the parameters of the graph generator and demonstrate how we generate the synthetic datasets with varying label homophily. We also summarize

the characteristics of the generated synthetic graphs that were used in Section 2.5 for the varying homophily and feature quality experiments. In Section 2.8.3, we summarize the hyperparameters of the models we used in this work. In Section 2.8.4, we provide the full original experiment results on all datasets reported in Micro- and Macro- F1, AUROC, and Average Precision score with the standard deviation of the 3 random splits if the dataset is not pre-splitted. Note that for higher precision, the scores are provided in percentages. Last but not least, we provide the motivation for using Average Precision in the main paper in Section 2.8.5.

2.8.1. BIOLOGICAL DATASET CONSTRUCTION

THE PROTEIN PHENOTYPE PREDICTION DATASET.

A phenotype is any observable characteristic or trait of a disease. Identifying the phenotypes associated with a particular protein could help in clinical diagnostics or finding possible drug targets.

To construct the phenotype prediction dataset, we first retrieve the experimentally validated protein-phenotype associations from the DisGeNET [47] database. We then (i) retain only those protein associations that are marked as "phenotype", (ii) match each disease to its first-level category in the MESH ontology [48], and (iii) remove any (phenotype) label with less than 100 associated proteins. To construct the edges, we acquire the protein functional interaction network from [49] (version 2020). We then (i) model each protein as a node in the graph, (ii) retain only the protein-protein interactions between the proteins that have the phenotype labels available, and (iii) remove any isolated nodes from the constructed graph. In the end, our dataset consists of $3,233$ proteins and $37,351$ edges. The node features are the 32-dimensional sequence-based embeddings retrieved from [50] and [51].

THE HUMAN PROTEIN SUBCELLULAR LOCATION PREDICTION DATASET (HUMLOC).

Proteins might exist at or move between different subcellular locations. Predicting protein subcellular locations can aid the identification of drug targets[1]. We retrieve the human protein subcellular location data from [45] which contains $3,106$ proteins. Each protein can have one to several labels in 14 possible locations. We then generate the graph multi-label node classification data as follows:

- We model each protein as a node in the graph. We retrieve the corresponding protein sequences from Uniprot [50]. We obtain the corresponding 32-dimensional node feature representation by feeding them to a pre-trained model [51] on protein sequences.
- Each node's label is the one-hot encoding (i.e., 14 dimensions) generated from its sub-cellular information. Each value in the label vector represents one sub-cellular location. A value of 1 indicates the corresponding protein exists at the respective location and 0 means otherwise.
- The edge information is generated from the protein-protein interactions retrieved from the IntAct [52] database. There exists a connection between two nodes in the graph if there exists an interaction between the corresponding proteins in IntAct. For each pair of proteins, more than one interaction of different types might exist. Therefore, we assign

[1] https://en.wikipedia.org/wiki/Protein_subcellular_localization_prediction

each edge a label. The edge label is modeled as a 21-dimensional vector where each value in the vector represents the confidence score for a particular connection type.

In the end, the HUMLOC dataset consists of 3,106 nodes and 18,496 edges. Each node can have one to several labels in the 14 possible locations. Among the 3,106 different proteins, 2,580 of them belong only to 1 location; 480 of them belong to 2 locations; 43 of them belong to 3 locations and 3 of them belong to 4 locations. Both the accession numbers and sequences are given. None of the proteins has more than 25% sequence identity to any other in the same subset (subcellular location). For a more detailed description of the original dataset, we refer the readers to [45].

THE EUKARYOTE PROTEIN SUBCELLULAR LOCATION PREDICTION DATASET (EUKLOC). We retrieve the eukaryote protein subcellular location multi-label data from [46]. We then employ the same data sources and pre-processing strategy as described for the HUMLOC dataset to generate the multi-label node classification dataset for eukaryote protein subcellular location prediction. In the end, the final pre-processed data contains 7,766 proteins (nodes) and 13,818 connections(edges). Each protein(node) can receive one to several labels in 22 possible locations.

2.8.2. PARAMETER STUDY OF THE GRAPH GENERATOR MODEL

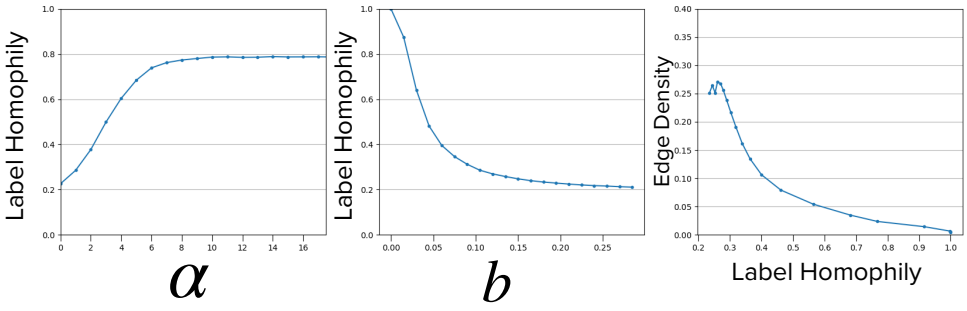


Figure 2.6: Visualization of the parameter study of the Graph Generator Model. The first two subplots demonstrate the relationships between the value of α and b and the label homophily of the generated synthetic datasets. The last subplot shows the edge density and the label homophily of the generated synthetic graphs. This shows with the same multi-label data, we can generate synthetic graphs with varying label homophily.

As mentioned in Section 2.4, the choice of α and b will directly determine the connection probability $p_{i,j}$ of each pair of nodes i and j and the homophily ratio of the generated synthetic graph. Here, we demonstrate how we choose the value of α and b to generate synthetic graphs with varying homophily ratios. Note that the valid range of α and b may differ when a different distance metric is used.

Firstly, we randomly choose 500 nodes from SYNTHETIC1 dataset and generate a series of small synthetic graphs with varying α and b and observe the relationship between them. Since we have two hyperparameters' ranges to determine, we first fix the value of one and explore the range of the other and then vice versa. To recall, b indicates the characteristic

distance at which $p_{ij} = \frac{1}{2}$, and our hamming distance should be in the range of $[0,1]$, we first fix b to 0.05. We chose a small value of b because the larger value of b would dominate the influence of α and the relationship between the homophily ratio and the change of the value of α becomes unclear. The relationship between the homophily ratio of the generated synthetic graphs and the value of α is shown in Figure 2.6(a). As shown in the subplot, the label homophily increases monotonically as the value of α increases in the range of $[0,10]$. As α is interpreted as the homophily parameter in [43], only positive values make sense.

Similarly, we then fix α to its middle value in the valid range, i.e. 5, and explore the valid range of b and visualize the relationship of the graph level homophily ratio and the value of b in Figure 2.6(b). As illustrated in the subfigure, the label homophily decreases as the b increases in the range of $(0,0.25]$. As b increases, the node pairs with bigger distance would also have 50% of the probability to be connected, the number of edges will increase, and the label homophily ratio will decrease. As b decreases, the node pair with a smaller distance would only have 50% of the probability of being connected. The model becomes cautious about connecting a node pair. The number of edges decreases and only the node pairs, which are alike will be connected, thus, the label homophily ratio increases.

Then, we use combinations of α and b to generate synthetic graphs with specific homophily ratios. We sample 20 αs and bs uniformly from their valid ranges with the increments 0.5 and 0.0125. Since the graph label homophily has an inverse linear relationship with α and b, we arrange the sampled b in reverse order and then form 20 value pairs (alpha, b). We generate 20 synthetic graphs from the multi-label dataset corresponding to SYNTHETIC1 with these value pairs (α, b) and plot the homophily ratio and the edge density of the generated graphs in Figure 2.6(c). As shown in the subplot, using the same multi-label data, we are able to generate synthetic graphs with varying homophiles. And the edge density decreases when label homophily increases. As in higher label homophily graphs, the generator will only connect the nodes that are highly similar to each other. In contrast, when the label homophily is low, the graph generator will connect every possible node pair in the graph resulting in denser graphs.

STATISTICS OF THE SYNTHETIC DATASETS

Here we summarize the characteristics of the generated synthetic graphs with varying label homophily and feature quality. The first row denotes the name of the synthetic graphs, where in the varying homophily experiments, the variants of datasets are named with their label homophily. The SYNTHETIC1 dataset is used in the varying feature quality experiment, where we remove the relevant features to create dataset variants with varying feature quality levels. Note that for all the datasets the label distribution stays the same as they are just different graphs generated from the same multi-label dataset. The statistics on label distribution for these datasets are given in Table 2.5 and Table 2.6.

2.8.3. HYPERPARAMETER SETTING

In this section, we summarize all the hyperparameters we used for the experiment section. The detailed setting is listed in Table 2.7 and 2.8.

More specifically, for MLP and all GNN-based methods, we summarize the number of layers, the dimension of the hidden layer, the learning rate, the patience of the Earlystopping, the weight decay, and the number of neighbors we sample for the models that require

Table 2.5: The number of edges and clustering coefficient of the synthetic datasets with varying label homophily and SYNTHETIC1. The row of '|E|' denotes the number of edges and the 'clustering coefficient' denotes the clustering coefficient of these datasets

Dataset	0.2	0.4	0.6	0.8	1.0	SYNTHETIC1
|E|	2.37M	598k	298k	79.5k	47.6k	1.00M
Clustering Coefficient	0.53	0.37	0.39	0.49	0.93	0.57

Table 2.6: The label distribution of the synthetic dataset used in this work. The column notations are same as in Table 2.1.

	$\|L\|$	$\|L_{med}\|$	$\|L_{mean}\|$	$\|L_{max}\|$	25%	50%	75%
label distribution	20	3	3.23	12	1	3	5

sampling.

We use the same number of layers and the same hidden size for MLP and the other GNN-based methods. The learning rate for the synthetic datasets in the varying feature quality and homophily experiments is 0.001 instead of 0.01 as in the other models because the performance of the H2GCN is further improved. We also use Earlystopping with the patience of 100 epochs to train the models properly. For GRAPHSAGE, we sample 25 and 20 one and two hops away neighbors for aggregation. As other GNN-based baselines do not use the sampling method, the corresponding cells are filled with "No".

For the only random-walk-based method, we deploy the default setting for all the datasets in this work as DEEPWALK already shows competitive performance. We perform 10 random walks with the walk length of 10 for each node to generate the sequence and use the window size of 5 for the training pairs, the generated embedding size is 64.

2.8.4. THE EXPERIMENT RESULTS REPORTED IN FOUR METRICS

We summarize the experimental results on the real-world datasets and the synthetic datasets in Table 2.9, Table 2.10, and Table 2.11, respectively. For better precision, we report the scores in percentages. Specifically, for the scores reported on OGB-PROTEINS, the difference between our results and those reported in the benchmark is because 1) we use 2 layer MLP without Node2Vec features. 2) We use Earlystopping with the patience of 100 epochs to prevent the models from being overtrained. 3) We use sampled local neighborhoods to have a consistent setting for all the datasets using GRAPHSAGE. The specific parameters we used are summarized in 2.8.3. Another pre-processing we did was to remove the isolated nodes in PCG. The details are described in Appendix 2.8.1.

Table 2.7: The hyperparameter setting for MLP and GNN baselines in this work for all datasets

	MLP	GCN, GAT, GraphSAGE	H2GCN	GCN-LPA
Layers	2	2	2	2 GCN 5 LPA
Hidden size	256	256	256	256
Learning rate	0.01	0.01	real-world: 0.01 synthetic: 0.001	0.01
Earlystopping patience	100	100	100	100
Weight decay	5e-4	5e-4	5e-4	1e-4
Sample for aggregation	No	GraphSAGE:[25, 10]	No	No

Table 2.8: The hyperparameter setting for DEEPWALK in this work for all datasets

	Number of walks	Walk length	Embedding size	Window size
DeepWalk	10	10	64	5

Table 2.9: Multi-label Node Classification results on real-world datasets. The results of BLOGCAT, YELP, PCG, and DBLP are the mean of three random splits with 60% for train-, 20% validation and 20% test dataset, while the results of OGB-PROTEINS, HUMLOC, EUKLOC are reported with the built-in split.

DATASET	METHOD	MICRO-F1	MACRO-F1	AUC-ROC	AP
BLOG-CAT	MLP	17.11 ± 0.64	2.49 ± 0.18	50.30 ± 1.04	4.25 ± 0.63
	DEEPWALK	$\mathbf{35.59 \pm 0.21}$	$\mathbf{19.74 \pm 0.54}$	$\mathbf{73.20 \pm 0.58}$	$\mathbf{18.55 \pm 0.17}$
	LANC	13.95 ± 2.02	4.55 ± 0.82	52.34 ± 0.91	$\underline{5.03 \pm 0.07}$
	GCN	16.69 ± 0.47	$\underline{2.63 \pm 0.08}$	47.85 ± 0.06	3.69 ± 0.04
	GAT	$\underline{17.22 \pm 0.52}$	2.48 ± 0.08	50.88 ± 1.45	4.05 ± 0.09
	GRAPHSAGE	16.18 ± 0.31	2.38 ± 0.27	$\underline{52.73 \pm 0.82}$	4.50 ± 0.12
	H2GCN	16.86 ± 0.34	2.60 ± 0.16	49.83 ± 1.08	3.92 ± 0.05
	GCN-LPA	17.15 ± 0.68	2.55 ± 0.19	51.35 ± 0.67	4.33 ± 0.31

Table 2.9: Multi-label Node Classification results on real-world datasets. The results of BLOGCAT, YELP, PCG, and DBLP are the mean of three random splits with 60% for train-, 20% validation and 20% test dataset, while the results of OGB-PROTEINS, HUMLOC, EUKLOC are reported with the built-in split.

DATASET	METHOD	MICRO-F1	MACRO-F1	AUC-ROC	AP
YELP	MLP	26.04 ± 0.09	18.55 ± 0.20	50.17 ± 0.01	9.58 ± 0.01
	DEEPWALK	49.78 ± 0.07	24.98 ± 0.04	50.67 ± 0.08	9.60 ± 0.02
	LANC	–	–	–	–
	GCN	52.21 ± 0.07	27.60 ± 0.04	53.81 ± 0.13	13.14 ± 0.06
	GAT	51.24 ± 0.08	26.66 ± 0.06	67.80 ± 0.05	15.00 ± 0.07
	GRAPHSAGE	**56.06 ± 0.10**	**31.26 ± 0.10**	**81.05 ± 0.25**	**25.09 ± 0.31**
	H2GCN	54.12 ± 0.01	30.52 ± 0.11	75.25 ± 0.46	22.57 ± 0.51
	GCN-LPA	50.31 ± 0.29	25.68 ± 0.43	61.09 ± 2.27	11.62 ± 0.74
OGB-PROTEINS	MLP	2.55	2.40	54.05	2.59
	DEEPWALK	**2.88**	**2.75**	68.75	4.41
	LANC	2.35	2.21	68.03	4.48
	GCN	2.77	2.63	**71.48**	**5.36**
	GAT	2.55	2.40	50.64	2.14
	GRAPHSAGE	2.59	2.43	55.83	2.68
	H2GCN	2.55	2.39	62.75	3.61
	GCN-LPA	2.56	2.41	53.22	2.33
DBLP	MLP	42.14 ± 0.27	32.04 ± 0.65	54.47 ± 0.07	34.97 ± 0.14
	DEEPWALK	63.27 ± 0.34	59.11 ± 0.36	74.81 ± 0.16	58.49 ± 0.25
	LANC	81.93 ± 0.29	80.39 ± 0.42	91.76 ± 0.36	83.55 ± 0.98
	GCN	**87.03 ± 0.20**	**85.80 ± 0.38**	94.15 ± 0.16	**89.27 ± 0.24**
	GAT	83.06 ± 0.17	81.26 ± 0.14	92.57 ± 0.07	82.93 ± 0.16
	GRAPHSAGE	85.22 ± 0.23	83.89 ± 0.21	**94.32 ± 0.02**	86.84 ± 0.18
	H2GCN	83.99 ± 0.92	82.56 ± 0.86	92.14 ± 0.57	85.82 ± 0.64
	GCN-LPA	82.88 ± 0.31	81.31 ± 0.34	90.17 ± 0.43	80.07 ± 1.24
PCG	MLP	38.04 ± 1.20	18.03 ± 1.29	51.07 ± 0.63	14.78 ± 0.60
	DEEPWALK	**42.26 ± 1.37**	**31.49 ± 0.90**	**63.58 ± 0.87**	**22.86 ± 1.00**
	LANC	36.28 ± 0.34	20.50 ± 1.15	56.58 ± 0.69	18.53 ± 1.14
	GCN	41.46 ± 1.21	25.59 ± 0.92	59.54 ± 0.82	21.03 ± 0.34
	GAT	36.91 ± 1.75	19.24 ± 0.75	56.33 ± 4.64	16.75 ± 2.17
	GRAPHSAGE	38.89 ± 1.17	24.44 ± 1.74	58.57 ± 0.08	18.45 ± 0.29
	H2GCN	39.05 ± 0.99	24.38 ± 2.17	58.10 ± 0.14	19.19 ± 0.49
	GCN-LPA	39.57 ± 1.12	22.90 ± 1.33	54.74 ± 0.95	16.71 ± 0.14
HUM-LOC	MLP	42.12	18.04	66.04	16.95
	DEEPWALK	45.26	23.30	65.67	18.58
	LANC	39.25	11.51	59.65	13.24
	GCN	**51.67**	**25.57**	67.28	**25.15**
	GAT	47.10	17.49	**72.47**	23.75

Table 2.9: Multi-label Node Classification results on real-world datasets. The results of BLOGCAT, YELP, PCG, and DBLP are the mean of three random splits with 60% for train-, 20% validation and 20% test dataset, while the results of OGB-PROTEINS, HUMLOC, EUKLOC are reported with the built-in split.

DATASET	METHOD	MICRO-F1	MACRO-F1	AUC-ROC	AP
	GRAPHSAGE	48.05	21.22	70.30	23.42
	H2GCN	45.39	18.35	64.31	17.23
	GCN-LPA	45.73	18.15	62.40	14.96
EU-K-LOC	MLP	43.58	11.13	66.83	12.00
	DEEPWALK	34.67	6.74	56.12	7.58
	LANC	36.08	4.55	51.13	6.16
	GCN	**45.86**	**12.27**	70.53	**15.15**
	GAT	41.58	6.76	**71.65**	13.59
	GRAPHSAGE	44.65	11.96	69.04	12.44
	H2GCN	44.93	11.80	69.45	13.35
	GCN-LPA	36.72	5.93	56.65	7.45

2.8.5. CHALLENGE OF THE EVALUATION METRICS

The Area under the ROC curve (AUC) and the Area under the Precision-Recall curve (AUPR) are two widely accepted non-parametric measurement scores used by existing works. Nevertheless, as discussed in [53], the AUC score is sometimes misleading for highly imbalanced datasets like those in our work. In addition, AUPR might lead to over-estimation of the models' performance when the number of thresholds (or unique prediction values) is limited [54]. For such reasons, as suggested in [54], we instead use the Average Precision (AP) score as our primary evaluation metric. Following previous works, we also report the F1 score. As it is a threshold-based metric we emphasize that it might be biased when the benchmarked models have different prediction score ranges.

2.8.6. CROSS-CLASS NEIGHBORHOOD SIMILARITY PLOTS

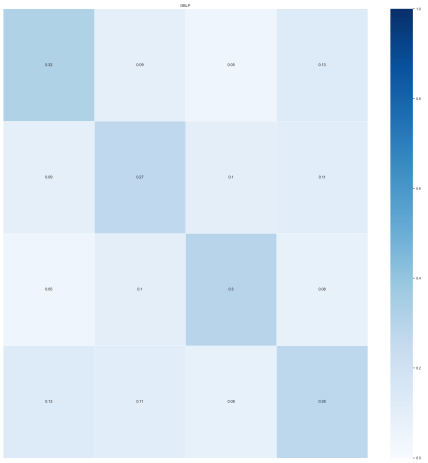
In this section, we put the heat maps of the cross-class neighborhood similarity for all the datasets used in this work. We use color coding, where a darker shade in the cell indicates a stronger cross-class neighborhood similarity.

Table 2.10: Multi-label Node Classification results on Synthetic dataset with varying feature quality. All results are the mean of three random splits. The Ratios of the relevant and the irrelevant features are $[0, 0.2, 0.5, 0.8, 1.0]$.

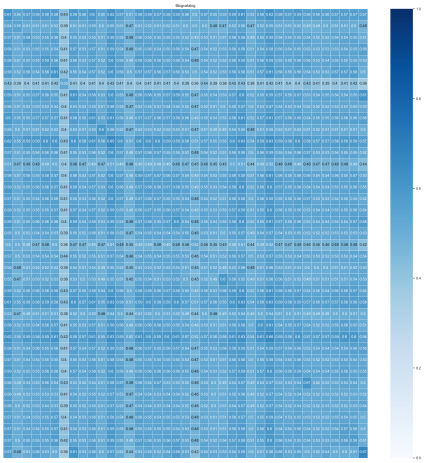
FEATURE RATIOS	METHOD	MICRO-F1	MACRO-F1	AUC-ROC	AP
0	MLP	67.05 ± 0.91	25.15 ± 0.89	50.99 ± 1.11	17.17 ± 0.42
	DEEPWALK	**86.22 ± 0.39**	**47.80 ± 0.31**	**84.15 ± 0.56**	**48.70 ± 0.51**
	LANC	69.67 ± 0.26	26.68 ± 1.16	75.36 ± 1.17	33.68 ± 0.88
	GCN	67.09 ± 0.95	25.12 ± 1.09	77.17 ± 0.92	31.27 ± 0.66
	GAT	66.67 ± 0.71	25.09 ± 1.24	61.96 ± 3.97	31.10 ± 3.17
	GRAPHSAGE	67.90 ± 1.40	25.87 ± 1.51	66.47 ± 0.31	30.01 ± 0.71
	H2GCN	71.12 ± 1.26	27.45 ± 1.18	80.83 ± 0.98	37.64 ± 1.72
	GCN-LPA	70.26 ± 2.00	26.70 ± 1.47	70.58 ± 2.04	33.65 ± 2.61
0.2	MLP	68.37 ± 0.55	26.61 ± 0.47	54.47 ± 0.19	18.70 ± 0.69
	DEEPWALK	**86.22 ± 0.39**	**47.80 ± 0.31**	**84.15 ± 0.56**	**48.70 ± 0.51**
	LANC	70.84 ± 1.75	27.62 ± 0.74	74.42 ± 2.02	34.24 ± 1.42
	GCN	67.42 ± 1.07	25.31 ± 1.09	77.47 ± 0.87	31.59 ± 0.63
	GAT	67.07 ± 1.07	25.03 ± 1.16	64.55 ± 0.60	33.90 ± 0.30
	GRAPHSAGE	70.14 ± 0.60	27.99 ± 1.07	69.44 ± 1.09	32.84 ± 0.79
	H2GCN	73.21 ± 1.07	28.85 ± 1.23	81.78 ± 1.24	40.12 ± 4.24
	GCN-LPA	69.67 ± 1.67	26.14 ± 1.53	71.02 ± 1.71	33.33 ± 1.31
0.5	MLP	68.83 ± 0.78	27.57 ± 1.10	61.63 ± 0.74	21.95 ± 0.40
	DEEPWALK	**86.22 ± 0.39**	**47.80 ± 0.31**	84.15 ± 0.56	**48.70 ± 0.51**
	LANC	73.21 ± 2.79	30.02 ± 2.01	77.05 ± 1.16	36.45 ± 1.30
	GCN	67.55 ± 1.08	25.39 ± 1.08	77.25 ± 0.85	31.14 ± 0.61
	GAT	67.76 ± 1.75	25.49 ± 1.46	64.40 ± 1.64	32.92 ± 0.88
	GRAPHSAGE	74.70 ± 0.49	31.25 ± 0.89	76.24 ± 0.47	37.65 ± 0.65
	H2GCN	76.16 ± 0.47	32.85 ± 0.28	**84.96 ± 0.55**	42.70 ± 0.27
	GCN-LPA	72.11 ± 1.89	27.80 ± 0.77	74.39 ± 1.26	36.84 ± 1.02
0.8	MLP	70.17 ± 0.81	29.37 ± 0.99	69.53 ± 0.46	27.65 ± 0.77
	DEEPWALK	**86.22 ± 0.39**	**47.80 ± 0.31**	84.15 ± 0.56	**48.70 ± 0.51**
	LANC	72.67 ± 3.56	29.67 ± 3.13	74.73 ± 1.07	35.32 ± 2.06
	GCN	69.56 ± 1.14	25.79 ± 1.14	77.03 ± 0.99	30.10 ± 0.71
	GAT	67.93 ± 0.75	25.40 ± 1.17	66.41 ± 3.55	33.78 ± 0.96
	GRAPHSAGE	75.66 ± 0.51	32.39 ± 1.50	78.31 ± 0.49	39.25 ± 0.86
	H2GCN	78.68 ± 0.38	36.17 ± 0.56	**86.01 ± 0.64**	44.21 ± 0.31
	GCN-LPA	72.62 ± 1.46	27.74 ± 0.67	73.99 ± 1.06	36.28 ± 1.57
1.0	MLP	72.90 ± 0.27	31.52 ± 0.52	75.23 ± 0.63	34.29 ± 0.07
	DEEPWALK	**86.22 ± 0.39**	**47.80 ± 0.31**	84.15 ± 0.56	**48.70 ± 0.51**
	LANC	74.13 ± 1.61	30.38 ± 1.29	75.26 ± 0.93	37.47 ± 0.27
	GCN	70.74 ± 0.80	26.57 ± 0.97	79.22 ± 0.82	33.70 ± 0.79
	GAT	70.22 ± 1.27	26.35 ± 0.93	66.52 ± 1.03	36.01 ± 0.72
	GRAPHSAGE	78.71 ± 0.59	35.77 ± 1.40	80.77 ± 0.17	43.05 ± 0.22
	H2GCN	79.97 ± 0.19	38.73 ± 0.91	**87.09 ± 0.12**	46.71 ± 0.37
	GCN-LPA	76.28 ± 1.27	31.91 ± 1.07	76.38 ± 0.36	39.10 ± 2.03

Table 2.11: Multi-label Node Classification results on Synthetic dataset with varying label homophily. All results are the mean of three random splits. The label homophiles (rounded up) are [0.2, 0.4, 0.6, 0.8, 1.0]

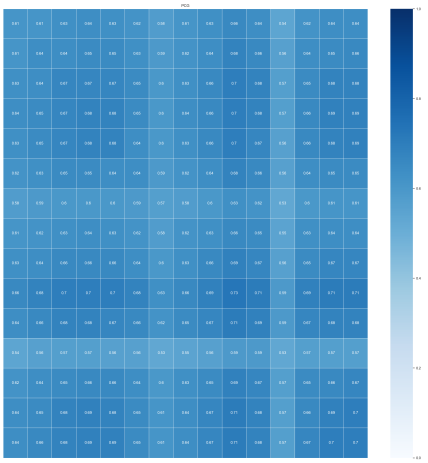
LABEL HOMOPHILY	METHOD	MICRO-F1	MACRO-F1	AUC-ROC	AP
0.2	MLP	72.90 ± 0.27	31.52 ± 0.52	**75.23 ± 0.63**	**34.29 ± 0.07**
	DEEPWALK	66.62 ± 0.62	26.02 ± 0.55	52.19 ± 0.91	18.07 ± 0.71
	LANC	67.05 ± 0.92	25.12 ± 2.20	54.49 ± 0.53	18.95 ± 0.43
	GCN	67.28 ± 0.67	25.21 ± 1.04	66.59 ± 0.58	26.06 ± 1.07
	GAT	65.09 ± 0.89	24.61 ± 1.41	51.43 ± 0.33	17.05 ± 0.36
	GRAPHSAGE	73.57 ± 0.20	31.22 ± 0.65	71.72 ± 0.30	28.94 ± 1.09
	H2GCN	**73.80 ± 0.59**	**31.86 ± 0.89**	73.24 ± 0.08	29.69 ± 0.53
	GCN-LPA	67.02 ± 0.89	25.24 ± 1.31	49.92 ± 0.93	17.02 ± 0.31
0.4	MLP	72.90 ± 0.27	31.52 ± 0.52	75.23 ± 0.63	34.29 ± 0.07
	DEEPWALK	**88.79 ± 0.20**	**54.74 ± 1.73**	85.74 ± 0.33	**52.15 ± 1.41**
	LANC	75.16 ± 0.77	33.69 ± 0.94	76.53 ± 0.86	37.99 ± 0.74
	GCN	72.16 ± 1.70	26.95 ± 1.18	80.19 ± 0.75	34.29 ± 0.88
	GAT	71.50 ± 1.30	27.58 ± 1.11	69.96 ± 1.69	35.86 ± 0.59
	GRAPHSAGE	79.09 ± 0.33	36.05 ± 1.56	81.00 ± 0.35	42.57 ± 0.49
	H2GCN	81.30 ± 0.25	40.54 ± 0.74	**87.91 ± 0.04**	48.36 ± 0.44
	GCN-LPA	78.11 ± 1.38	32.71 ± 1.47	78.00 ± 0.70	40.80 ± 0.20
0.6	MLP	72.90 ± 0.27	31.52 ± 0.52	75.23 ± 0.63	34.29 ± 0.07
	DEEPWALK	**95.94 ± 0.05**	**82.58 ± 0.19**	**95.32 ± 0.80**	**81.34 ± 0.95**
	LANC	80.36 ± 1.81	39.65 ± 3.14	81.85 ± 2.66	43.42 ± 3.61
	GCN	75.47 ± 0.46	29.43 ± 0.43	84.08 ± 0.72	38.78 ± 0.33
	GAT	75.13 ± 0.39	29.26 ± 0.56	72.85 ± 0.79	39.01 ± 0.23
	GRAPHSAGE	82.46 ± 0.67	40.46 ± 1.67	83.24 ± 0.89	45.79 ± 0.91
	H2GCN	83.40 ± 1.94	44.32 ± 1.59	89.98 ± 0.86	51.19 ± 1.92
	GCN-LPA	85.45 ± 2.42	47.67 ± 4.72	83.96 ± 2.22	49.54 ± 3.54
0.8	MLP	72.90 ± 0.27	31.52 ± 0.52	75.23 ± 0.63	34.29 ± 0.07
	DEEPWALK	**96.53 ± 0.61**	**89.25 ± 1.96**	**95.81 ± 0.47**	**86.93 ± 1.92**
	LANC	79.35 ± 0.97	46.03 ± 2.24	87.75 ± 0.60	48.10 ± 1.17
	GCN	80.72 ± 0.55	36.60 ± 1.31	86.82 ± 0.62	44.98 ± 0.40
	GAT	77.35 ± 0.49	31.63 ± 1.10	83.10 ± 0.69	42.82 ± 0.86
	GRAPHSAGE	84.22 ± 0.38	48.16 ± 1.42	87.37 ± 0.34	53.34 ± 0.84
	H2GCN	86.00 ± 2.63	55.85 ± 4.92	91.61 ± 1.64	57.17 ± 2.57
	GCN-LPA	88.96 ± 0.68	64.09 ± 4.18	89.53 ± 1.07	60.44 ± 4.41
1.0	MLP	72.90 ± 0.27	31.52 ± 0.52	75.23 ± 0.63	34.29 ± 0.07
	DEEPWALK	80.25 ± 0.11	62.80 ± 0.46	83.83 ± 1.08	55.16 ± 0.67
	LANC	83.37 ± 0.96	60.77 ± 3.51	92.53 ± 1.27	62.85 ± 3.01
	GCN	81.61 ± 0.25	42.19 ± 0.45	86.48 ± 0.31	49.32 ± 0.92
	GAT	79.37 ± 0.64	34.67 ± 1.87	87.20 ± 2.15	43.90 ± 2.90
	GRAPHSAGE	82.15 ± 0.49	46.06 ± 0.99	89.73 ± 0.45	55.25 ± 0.42
	H2GCN	**91.59 ± 1.74**	**76.09 ± 5.96**	**93.94 ± 1.03**	**65.20 ± 5.71**
	GCN-LPA	86.92 ± 1.13	60.71 ± 3.01	90.75 ± 0.46	58.28 ± 3.02



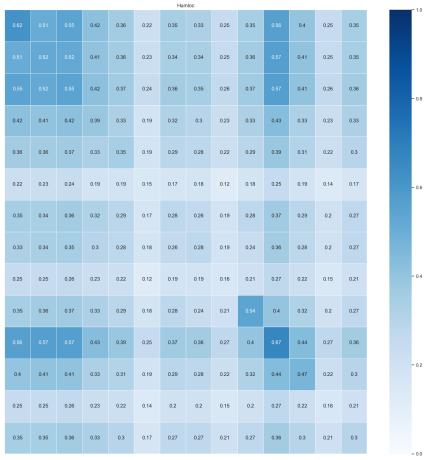
(a) Cross class Neighborhood Similarity in DBLP



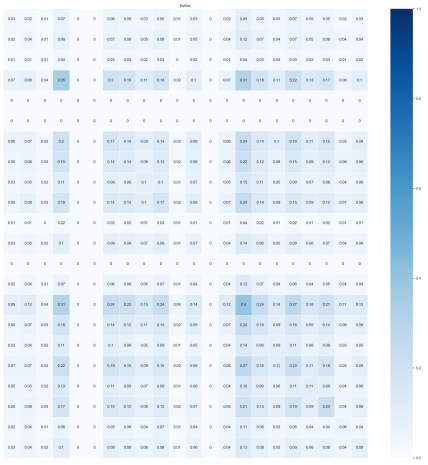
(b) Cross class Neighborhood Similarity in BLOGCAT



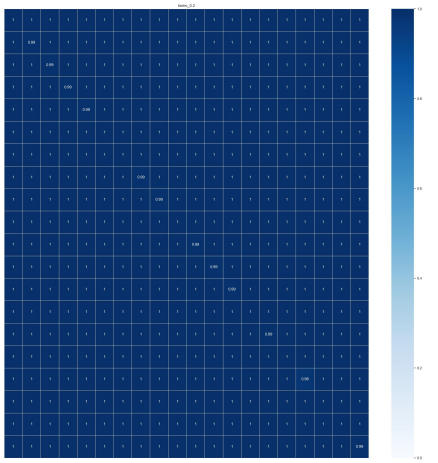
(c) Cross class Neighborhood Similarity in PCG



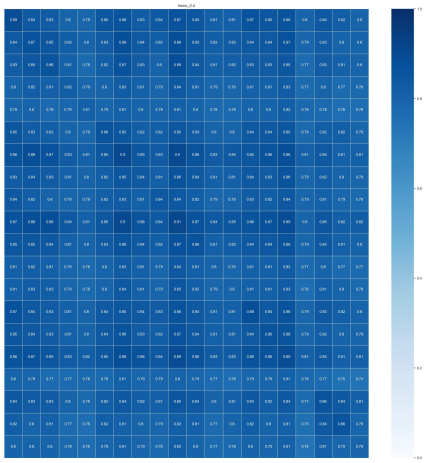
(d) Cross class Neighborhood Similarity in HUMLOC



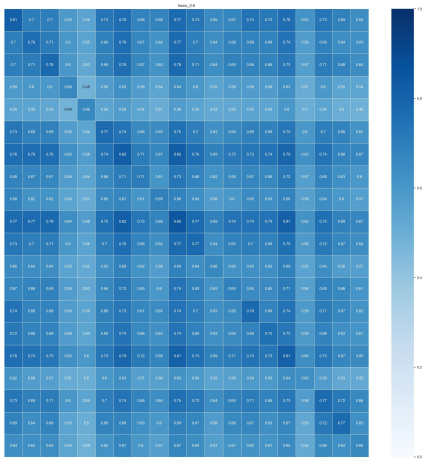
(e) Cross class Neighborhood Similarity in EUKLOC



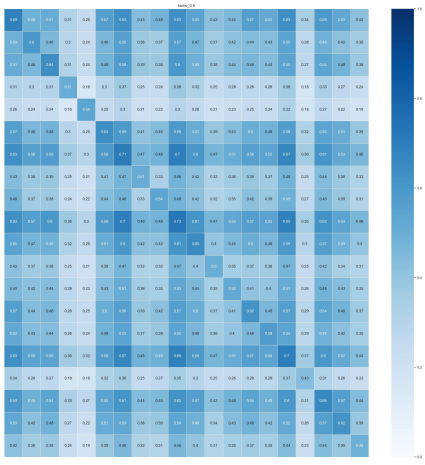
(a) Cross-class Neighborhood Similarity in graph label homophily=0.2



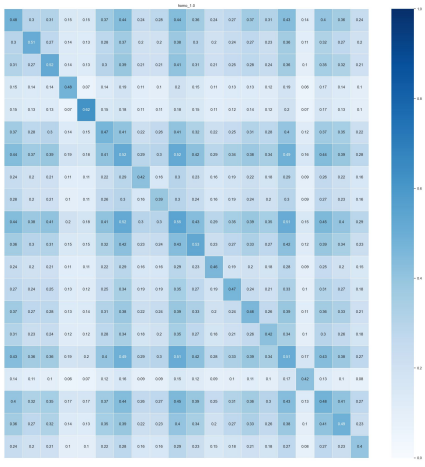
(b) Cross-class Neighborhood Similarity in graph label homophily=0.4



(c) Cross-class Neighborhood Similarity in graph label homophily=0.6



(d) Cross-class Neighborhood Similarity in graph label homophily=0.8



(e) Cross-class Neighborhood Similarity in graph label homophily=1.0

REFERENCES

1. Zhang, M.-L. & Zhou, Z.-H. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering* **26,** 1819–1837 (2013).
2. Ma, J., Chang, B., Zhang, X. & Mei, Q. CopulaGNN: towards integrating representational and correlational roles of graphs in graph neural networks. *arXiv preprint arXiv:2010.02089* (2020).
3. Jia, J. & Benson, A. R. *Residual correlation in graph neural network regression* in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020), 588–598.
4. Zhu, J. *et al.* Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. *Advances in Neural Information Processing Systems* **33** (2020).
5. Hu, W. *et al.* Open Graph Benchmark: Datasets for Machine Learning on Graphs. *arXiv preprint arXiv:2005.00687* (2020).
6. Liu, J., Chang, W.-C., Wu, Y. & Yang, Y. *Deep Learning for Extreme Multi-Label Text Classification* in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Association for Computing Machinery, Shinjuku, Tokyo, Japan, 2017), 115–124. ISBN: 978-1-4503-5022-8. https://doi.org/10.1145/3077136.3080834.
7. Song, D., Vold, A., Madan, K. & Schilder, F. Multi-Label Legal Document Classification: A Deep Learning-Based Approach with Label-Attention and Domain-Specific Pre-Training. *Inf. Syst.* **106.** ISSN: 0306-4379. https://doi.org/10.1016/j.is.2021.101718 (May 2022).
8. Wang, B. *et al.* Ranking-Based Autoencoder for Extreme Multi-label Classification. *CoRR* **abs/1904.05937.** arXiv: 1904.05937. http://arxiv.org/abs/1904.05937 (2019).
9. Huynh, D. & Elhamifar, E. *Interactive multi-label cnn learning with partial labels* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 9423–9432.
10. Jain, V., Modhe, N. & Rai, P. *Scalable Generative Models for Multi-label Learning with Missing Labels* in *Proceedings of the 34th International Conference on Machine Learning* (eds Precup, D. & Teh, Y. W.) **70** (PMLR, June 2017), 1636–1644. https://proceedings.mlr.press/v70/jain17a.html.
11. Chu, H.-M., Yeh, C.-K. & Wang, Y.-C. F. *Deep generative models for weakly-supervised multi-label classification* in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 400–415.
12. Hu, M., Han, H., Shan, S. & Chen, X. *Weakly supervised image classification through noise regularization* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 11517–11525.
13. Liu, W., Wang, H., Shen, X. & Tsang, I. W. The emerging trends of multi-label learning. *IEEE transactions on pattern analysis and machine intelligence* **44,** 7955–7974 (2021).
14. Perozzi, B., Al-Rfou, R. & Skiena, S. *Deepwalk: Online learning of social representations* in *Proc. of the International Conference on Knowledge Discovery and Data Mining* (2014).

15. Khosla, M., Leonhardt, J., Nejdl, W. & Anand, A. *Node representation learning for directed graphs* in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2019), 395–411.

16. Ou, M., Cui, P., Pei, J., *et al. Asymmetric transitivity preserving graph embedding* in *Proc. of the International Conference on Knowledge Discovery and Data Mining* (2016), 1105–1114.

17. Shi, M., Tang, Y., Zhu, X. & Liu, J. Multi-Label Graph Convolutional Network Representation Learning. *CoRR* **abs/1912.11757.** arXiv: 1912.11757. http://arxiv.org/abs/1912.11757 (2019).

18. Zhou, C. *et al.* Multi-label graph node classification with label attentive neighborhood convolution. *Expert Systems with Applications* **180,** 115063. ISSN: 0957-4174. https://www.sciencedirect.com/science/article/pii/S0957417421005042 (2021).

19. Song, Z., Meng, Z., Zhang, Y. & King, I. *Semi-supervised Multi-label Learning for Graph-structured Data* in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (2021), 1723–1733.

20. Kipf, T. N. & Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907* (2016).

21. Veličković, P. *et al.* Graph Attention Networks. *International Conference on Learning Representations.* accepted as poster. https://openreview.net/forum?id=rJXMpikCZ (2018).

22. Hamilton, W. L., Ying, R. & Leskovec, J. Inductive Representation Learning on Large Graphs. *CoRR* **abs/1706.02216.** arXiv: 1706.02216. http://arxiv.org/abs/1706.02216 (2017).

23. Yang, C., Liu, J. & Shi, C. *Extract the Knowledge of Graph Neural Networks and Go Beyond it: An Effective Knowledge Distillation Framework* in *Proceedings of The Web Conference 2021 (WWW '21)* (ACM, 2021).

24. Wang, H. & Leskovec, J. Unifying Graph Convolutional Neural Networks and Label Propagation. *CoRR* **abs/2002.06755.** arXiv: 2002.06755. https://arxiv.org/abs/2002.06755 (2020).

25. Saini, D. *et al. GalaXC: Graph Neural Networks with Labelwise Attention for Extreme Classification* in (Association for Computing Machinery, Ljubljana, Slovenia, 2021). ISBN: 978-1-4503-8312-7. https://doi.org/10.1145/3442381.3449937.

26. Lanchantin, J., Sekhon, A. & Qi, Y. *Neural message passing for multi-label classification* in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2019), 138–163.

27. Chen, Z.-M., Wei, X.-S., Wang, P. & Guo, Y. *Multi-label image recognition with graph convolutional networks* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), 5177–5186.

28. Li, I., Li, T., Li, Y., Dong, R. & Suzumura, T. Heterogeneous Graph Neural Networks for Multi-label Text Classification. *CoRR* **abs/2103.14620.** arXiv: 2103.14620. https://arxiv.org/abs/2103.14620 (2021).

29. Zong, D. & Sun, S. GNN-XML: Graph Neural Networks for Extreme Multi-label Text Classification. *CoRR* **abs/2012.05860.** arXiv: 2012.05860. https://arxiv.org/abs/2012.05860 (2020).

30. Ma, Q., Yuan, C., Zhou, W. & Hu, S. *Label-specific dual graph neural network for multi-label text classification* in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (2021), 3855–3864.

31. Zheng, X., Liang, X. & Wu, B. *Capsule Graph Neural Network for Multi-Label Image Recognition (Student Abstract)* in *Proceedings of the AAAI Conference on Artificial Intelligence* **36** (2022), 13117–13118.

32. Shi, M., Tang, Y., Zhu, X. & Liu, J. Multi-label graph convolutional network representation learning. *IEEE Transactions on Big Data* (2020).

33. Xu, L. *et al. Hierarchical Multi-label Text Classification with Horizontal and Vertical Category Correlations* in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, Nov. 2021), 2459–2468. https://aclanthology.org/2021.emnlp-main.190.

34. Cheng, Y., Ma, M., Li, X. & Zhou, Y. Multi-label classification of fundus images based on graph convolutional network. *BMC Medical Informatics and Decision Making* **21,** 1–9 (2021).

35. Pal, A., Selvakumar, M. & Sankarasubbu, M. Multi-label text classification using attention-based graph neural network. *arXiv preprint arXiv:2003.11644* (2020).

36. Ma, Y., Liu, X., Shah, N. & Tang, J. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134* (2021).

37. Tang, L. & Liu, H. *Relational Learning via Latent Social Dimensions* in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, Paris, France, 2009), 817–826. ISBN: 978-1-60558-495-9. https://doi.org/10.1145/1557019.1557109.

38. Zeng, H., Zhou, H., Srivastava, A., Kannan, R. & Prasanna, V. K. GraphSAINT: Graph Sampling Based Inductive Learning Method. *CoRR* **abs/1907.04931.** arXiv: 1907.04931. http://arxiv.org/abs/1907.04931 (2019).

39. Consortium, G. O. The gene ontology resource: 20 years and still going strong. **47,** D330–D338 (2018).

40. D, S. *et al.* STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Research* **47,** D607–D613 (2019).

41. Akujuobi, U., Han, Y., Zhang, Q. & Zhang, X. Collaborative Graph Walk for Semi-supervised Multi-Label Node Classification. *CoRR* **abs/1910.09706.** arXiv: 1910.09706. http://arxiv.org/abs/1910.09706 (2019).

42. Tomás, J. T., Spolaôr, N., Cherman, E. A. & Monard, M. C. A framework to generate synthetic multi-label datasets. *Electronic Notes in Theoretical Computer Science* **302,** 155–176 (2014).

43. Boguná, M., Pastor-Satorras, R., Díaz-Guilera, A. & Arenas, A. Models of social networks based on social distance attachment. *Physical review E* **70,** 056122 (2004).

44. Talaga, S. & Nowak, A. Homophily as a Process Generating Social Networks: Insights from Social Distance Attachment Model. *Journal of Artificial Societies and Social Simulation* **23,** 6. ISSN: 1460-7425 (2020).

45. Shen, H.-B. & Chou, K.-C. Hum-mPLoc: an ensemble classifier for large-scale human protein subcellular location prediction by incorporating samples with multiple sites. *Biochemical and biophysical research communications* **355,** 1006–1011 (2007).

46. Chou, K.-C. & Shen, H.-B. Euk-mPLoc: a fusion classifier for large-scale eukaryotic protein subcellular location prediction by incorporating multiple sites. *Journal of Proteome Research* **6,** 1728–1734 (2007).

47. Piñero, J. *et al.* The DisGeNET knowledge platform for disease genomics: 2019 update. *Nucleic acids research* **48,** D845–D855 (2020).

48. Bhattacharya, S., Ha- Thuc, V. & Srinivasan, P. MeSH: a window into full text for document summarization. *Bioinformatics* **27,** i120–i128 (2011).

49. Wu, G., Feng, X. & Stein, L. A human functional protein interaction network and its application to cancer data analysis. *Genome biology* **11,** 1–23 (2010).

50. Consortium, U. UniProt: a hub for protein information. *Nucleic acids research* **43,** D204–D212 (2015).

51. Yang, X., Yang, S., Li, Q., Wuchty, S. & Zhang, Z. Prediction of human-virus protein-protein interactions through a sequence embedding-based machine learning method. *Computational and structural biotechnology journal* **18,** 153–161 (2020).

52. Kerrien, S. *et al.* The IntAct molecular interaction database in 2012. *Nucleic acids research* **40,** D841–D846 (2012).

53. Yang, Y., Lichtenwalter, R. N. & Chawla, N. V. Evaluating link prediction methods. *Knowledge and Information Systems* **45,** 751–782 (2015).

54. Dong, T. N. & Khosla, M. *Towards a consistent evaluation of miRNA-disease association prediction models* in *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2020), 1835–1842.

3

GNN-MultiFix: Addressing the pitfalls of GNNs for multi-label node classification

[1]This chapter is based on the manuscript: **Tianqi Zhao**, Megha Khosla, *GNN-MultiFix: Addressing the Pitfalls of GNNs for Multi-Label Node Classification*, https://arxiv.org/abs/2411.14094, currently under review for the *Journal of Data-centric Machine Learning Research (DMLR)*.

3.1. INTRODUCTION

Graph neural networks (GNNs) have become highly effective models for representing graph-structured data, delivering state-of-the-art performance in various tasks. One such prototypical task is the node classification task, which involves predicting node labels based on the graph structure and a partially labeled node set. While contemporary works have demonstrated significant improvements in multi-class node classification, where each node has at most one label, a recent work [1] highlights the limitations of current approaches in addressing multi-label node classification, where each node can have multiple labels.

Specifically, classical GNNs [2–4] show superior performance on multi-class graph datasets with high label homophily, where connected nodes typically share the same labels. However, in multi-label datasets, a node usually shares only a small fraction of common labels with its neighbors, resulting in low label homophily per edge. Consequently, the aggregated information from the local neighborhood of each node contains information about diverse characteristics of the nodes in its local neighborhoods, making it challenging for the GNNs to differentiate the most informative part to infer the labels of the ego nodes.

While [1] showed that GNNs exploiting higher order neighborhoods are equally insufficient for tackling multi-label node, we provide an even more surprising result. In particular, we demonstrate that GNNs are significantly outperformed by a simple MAJORITYVOTE baseline, which relies solely on the label information of immediate neighbors from the training set to determine the label distribution of test nodes.

Taking a step further, we analyse the training dynamics of representative methods for multi-label node classification. We validate the insufficiency of GNNs to learn over multi-label datasets even in case of abundant training data. As an example, Figure 3.1 shows box plots of per-node training losses for GCN [2] when trained on all nodes of the multi-label dataset BLOGCAT [5]. The y-axis reports the binary cross-entropy (BCE) loss for individual nodes. We observe that, even when the mean training loss converges to 0, many atypical nodes still have high loss, indicating that GCN fails to learn their labels adequately.

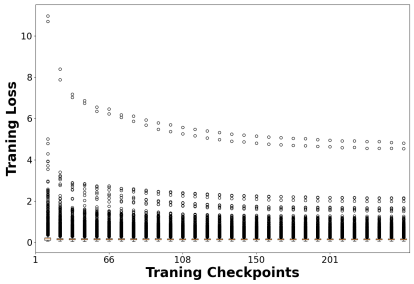


Figure 3.1: Training dynamics when GCN is trained using all nodes of BLOGCAT

The limitations of GNNs are usually studied in context of their abilities to recognize isomorphic graphs or distinguish non-isomorphic graphs. Several approaches [6] for feature-, topology- and GNN architecture enhancement have been proposed to improve the expressive power of GNNs. Nevertheless, these enhancements are usually motivated by the task of graph classification. While for graph classification, one would like to map structurally similar graphs to the similar representations, this may not always be true for the case of node-classification.

Consider for example a social network as shown in Figure 3.2, for which only the topological structure and the node labels are known. Two far away nodes (Alice and Bob) in such a graph may have similar local or even higher order topological structure (making their computational graphs as utilized by GNNs isomorphic) but can have completely different

interests (labels).

From the above discussion we make two key observations. *First*, the neighborhood aggregation approach of GNNs even of the ones which exploit higher order neighborhood is insufficient to encode node proximity information. While such an observation is not new, it has only been seen as a limitation for the task of link prediction [7] and has been overlooked for node classification. In fact the task of link prediction and node classification are closely related for certain network types. For example, in a social network new friendships (links) may lead to adoption of new user interests (labels). In other words links could be the *causal* factors for node labels. Several complex GNN architectures have been proposed to jointly encode positional and structural representations of the nodes. Nonetheless, most of the improvements are exhibited for the graph classification task and the nuances of node classification especially for the multi-label case are simply ignored. *Second*, GNNs are not able to exploit the label distribution of neighborhood nodes specifically in case of multi-label datasets. For instance, we show that a simple baseline which predicts a node labels based on a simple aggregation of known labels of its neighbors (in case they are in the training set) outperforms existing GNNs by a large margin.

Figure 3.2: Social network with similar neighborhoods but different labels.

Our Contributions. First, we conduct an empirical analysis of the training dynamics of existing GNN models on real world datasets, demonstrating their limited learning capabilities when applied to multi-label classification tasks. Second, we introduce a simple yet novel framework, GNN-MULTIFIX, designed to fully leverage the available input information for each node—namely its features, labels, and position in the graph. Through a combination of theoretical analysis and large-scale experiments, we demonstrate that GNN-MULTIFIX outperforms even highly expressive GNNs, which are designed to go beyond the limitations of the 1-Weisfeiler-Lehman (1-WL) test.

3.2. BACKGROUND AND RELATED WORK

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph where $\mathcal{V} = \{v_1, \cdots, v_n\}$ is the set of vertices, $\mathcal{E}$ represents the set of links/edges among the vertices. We further denote the adjacency matrix of the graph by $\mathbf{A} \in \{0,1\}^{n \times n}$ and $a_{i,j}$ denotes whether there is an edge between v_i and v_j. $\mathcal{N}(v)$ represents the immediate neighbors of node v in the graph. Furthermore, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{n \times D}$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n] \in \{0,1\}^{n \times C}$ represent the feature and label matrices corresponding to the nodes in $\mathcal{V}$. In the feature matrix $\mathbf{X}$ and the label matrix $\mathbf{Y}$, the i-th row $\mathbf{x}_i$ and $\mathbf{y}_i$ represent the feature vector and the multi-hot encoded label vector of node i, respectively. Let $\ell(i)$ denote the set of labels that are assigned to node i. Finally, let $\mathcal{F}$ correspond to the feature set and $\mathcal{L}$ be the set of all labels.

Problem Setting. In this work, we focus on the general setting of the node classification task on graph-structured data. In particular, we are given a set of labeled nodes $\mathcal{V}_l$ and unlabeled nodes $\mathcal{V}_u$ such that each node can have arbitrary number of labels. We are then interested in predicting labels of $\mathcal{V}_u$. We assume that the training nodes are completely labeled. We deal with the transductive setting multi-label node classification problem, where the features to all nodes $\mathbf{X}$, labels of the training nodes $\mathbf{Y}_{\mathcal{V}_l}$, and the graph structure $\mathcal{G}$ are present during training.

Graph Neural Networks and Node Classification. In the node classification task, classical GNNs [2–4, 8] usually take node features $\mathbf{X}$ and the adjacency matrix $\mathbf{A}$ as input. They compute node representation by recursive aggregation and transformation of feature representations of its neighbors which are then passed to a classification module subsequently generating the probability distribution matrix $\hat{\mathbf{Y}} = [\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_n] \in \mathbb{R}^{n \times C}$ across all labels for the input nodes: $\hat{\mathbf{Y}} = f_\theta(\mathbf{X}, \mathbf{A})$, where θ represents the set of parameters in the learned model. During the deployment of classical GNNs, training labels $\mathbf{Y}_{\mathcal{Y}_l}$ are solely utilized for supervision in calculating the task-specific loss $\mathscr{L}$. The function $\mathscr{L}$ is typically the cross-entropy loss defined as: $\mathscr{L} = -\sum_{i \in \mathcal{Y}_l} \mathbf{y}_i \log(\hat{\mathbf{y}}_i)$. For the multi-label node classification, a sigmoid layer is employed as the last layer to predict the class probabilities: $\mathbf{y}_i \leftarrow (\text{sigmoid}(\mathbf{z}_i \mathbf{W}))$, where $\mathbf{z}_i$ and $\mathbf{W}$ correspond to the node representation at the last layer and learnable weight matrix in the classification module respectively.

Definition 3 (Computational graph)**.** *In what follows we refer to the neighborhood graph of a node used by the GNN to obtain its feature representation as its computational graph.*

Definition 4 (Label Homophily)**.** *Given a multi-label graph $\mathcal{G}$, we define the label homophily h of $\mathcal{G}$ (following [1]) as the average of the Jaccard similarity of the label set of all connected nodes in the graph:* $r_{homo} = \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} \frac{|\ell(i) \cap \ell(j)|}{|\ell(i) \cup \ell(j)|}$.

3.2.1. RELATED WORK

Label-informed GNNs. Building on the success of classical GNNs, recent work has sought to exploit input label information beyond its conventional use in the training objective [9–11]. An early attempt involved applying label propagation to learn edge weights, enabling a weighted aggregation of local features for each node [10]. This approach allows nodes with similar labels to exchange information while suppressing the flow from nodes of dissimilar classes. However, the effectiveness of the resulting embeddings remains highly dependent on the quality of the input features. Another line of work augments node features with padded label vectors [12, 13], injecting label information more directly into the learning process.

A few studies have proposed methods specifically tailored for the multi-label node classification setting. For instance, LANC [11] learns label embeddings directly within the graph and incorporates them into the aggregated feature embeddings for each node. More recently, Sun *et al.* [14] have focused on leveraging label correlation information during representation learning. However, the effectiveness of such methods can be limited by the sparsity of the label correlation matrix, which may hinder the ability of the model to capture meaningful inter-label dependencies.

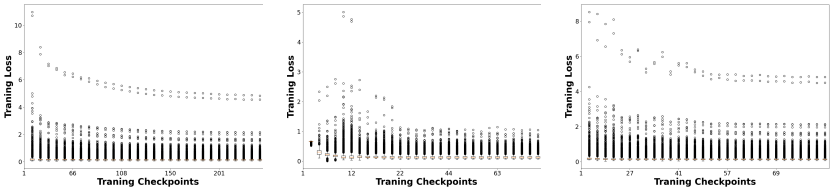
Other works [15–20] focusing on using GNNs to exploit the relations between the data and labels in the context of multi-label learning with independent euclidean data, such as text and images are out of the scope of current work.

Positional Encoding in GNNs. Positional encoding (PE) has emerged as a key component in enhancing the expressive power of GNNs, particularly for distinguishing structurally similar or symmetric nodes. One class of methods, such as Laplacian Positional Encoding, leverages the eigenvectors of the graph Laplacian to encode the relative position of a node within the global graph structure [21–23]. In contrast, learnable positional encoding methods aim to dynamically capture node identity or positional information during training. For

example, GNN-LSPE [21] augments input features with structural features and integrates them into message passing layers to produce position-aware node embeddings. Similarly, ID-GNN-FAST [24] enriches node representations by introducing unique ID embeddings for each node.

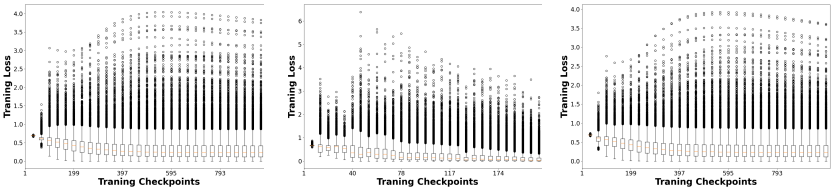
Data hardness analysis. A few works [25–27] have also focused on characterizing the data hardness for machine learning models. For example, [25] proposed a instance-level difficulty evaluation metric and rank the difficulty of each input data for analysis. [26] curate subsets from the input datasets and monitor the training dynamics on the them. However, to our best knowledge, none of these works have focused on graph-structured data and GNNs.

3.3. ANALYSIS OF TRAINING DYNAMICS OF GNNS



(a) GCN **(b)** H2GCN **(c)** ID-GNN-FAST

Figure 3.3: Visualization of training dynamics of GCN, H2GCN, and ID-GNN-FAST on the dataset BLOGCAT. The caption of each subfigure indicates the name of the model.



(a) GCN **(b)** H2GCN **(c)** ID-GNN-FAST

Figure 3.4: Visualization of training dynamics of GCN, H2GCN, and ID-GNN-FAST on the dataset DBLP. The caption of each subfigure indicates the name of the model.

In order to support our arguments regarding the limitations of existing methods for multi-labeled nodes, we analyze the training dynamics of GCN, ID-GNN-FAST, and H2GCN, three distinct approaches, each optimized for different types of graph data. GCN performs well on high-homophily, multi-class datasets, where connected nodes tend to belong to the same class. In contrast, H2GCN is designed to be robust to the changes of levels of homophily with the most significant improvement shown in heterophilic multi-class graphs, where neighboring nodes often differ in class labels, addressing the limitations of

classical GNNs. Finally, ID-GNN-FAST incorporates positional information by injecting identity-aware information into the input features.

We chose two datasets with contrasting homophily levels: BLOGCAT (low homophily) and DBLP (high homophily). We visualized the loss dynamics during training on the training data at the saved checkpoints for the selected models. The x-axis denotes the epoch indices where checkpoints were saved, while the y-axis represents the distribution of losses for each training node. We sampled 30 uniformly spaced checkpoints throughout the training process. We make the following observations about the expressive power of various methods.

ID-GNN-FAST does not help training behavior any more than GCN We consider the extreme case of BLOGCAT where no node features are available. Further, we label all nodes in BLOGCAT and use them for training GCN and ID-GNN-FAST. In Figure 3.3, both GCN and ID-GNN-FAST underfits the training data where the training loss does not converge for a large number of nodes. As in the absence of node features node representations are learned based on local neighborhood structures, it is expected that two nodes with similar neighborhood structures will be mapped to similar representations irrespective of their positions in the network. In such a case, the expressive power of such models cannot be arbitrarily trivially by increasing the number of learnable parameters. The situation does not improve when using ID-GNN-FAST which supposedly encodes additionally the positional information of the nodes.

GNNs tend to focus on easier nodes while neglecting the harder ones. In Figure 3.4, we compare the training dynamics of GCN, H2GCN, and ID-GNN-FAST when trained on 60% of the labeled nodes of DBLP. For all three models, we observe that although the training losses decrease and converge more effectively than on BLOGCAT, there still remains a significant number of nodes with much higher losses than the average. Notably, the training losses for the hard nodes increase as the average training loss decreases, indicating that despite the high homophily of the dataset the models are still unable to learn on a significant number of nodes.

3.4. OUR PROPOSED METHOD

We propose a simple yet effective method that maximizes the utility of the information in the graph, even when parts of the input are of poor quality. Our model, which we refer to as GNN-MULTIFIX, consists of three input processing modules, namely (i) feature representation module, (ii) label representation module (iii) node position/proximity representation module, each capturing different and independent aspects of a node representation. We summarize the algorithm in Appendix 3.8.1 and explain the design of each modules in the following:

The Feature Representation Module. The feature representation module takes as input the node feature matrix $\mathbf{X}$ and the adjacency matrix $\mathbf{A}$ and iterates for K layers and outputs node feature representation $\mathbf{h}_{f_v}^{(K)}$ for each node v as:

$$\mathbf{z}_{f_v}^{(K)} = \text{AGG}_{(f)}^{(K)}\left(\left\{\mathbf{h}_{f_u}^{(K-1)} : u \in \mathcal{N}(v)\right\}\right), \mathbf{h}_{f_v}^{(K)} = \text{COMB}_{(f)}^{(K)}\left(\mathbf{z}_{f_v}^{(K)}, \mathbf{h}_{f_v}^{(K-1)}\right) \tag{3.1}$$

where $\mathbf{h}_{f_v}^{(0)}$ is initialized as $\mathbf{h}_{f_v}^{(0)} = \mathbf{x}_v$ and AGG and COMB are general purpose aggregation and combination functions defined over one-hop node neighborhood. One can in general instantiate these functions corresponding to any of the existing message passing GNNs. Our method can then be seen as enhancing any existing GNN with our additional label and positional representations modules. Inspired by the observation [1] that in multi-label graphs, despite low homophily, similar nodes are often located in the local neighborhoods we limit the feature aggregation in each layer to one-hop neighborhood.

The Label Representation Module. Secondly, we design a label representation module to generate label-informed embeddings for each node by partially leveraging the ground-truth labels of the training nodes. For each labeled node $v \in \mathcal{V}_l$, we initialize its label representation as $\mathbf{h}_{l_v}^{(0)} = \mathbf{y}_v$, where $\mathbf{y}_v \in \{0,1\}^C$ is a multi-hot vector indicating the presence of each label (i.e., the i-th entry $\mathbf{y}_v^{(i)} = 1$ if node v has label i, and 0 otherwise). For unlabeled nodes $v \notin \mathcal{V}_l$, we initialize $\mathbf{h}_{l_v}^{(0)}$ with a padding vector $\mathbf{p} \in \mathbb{R}^C$, which may have each entry equal to $1/|C|$ or zero. Additionally, we define a label influence matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, where $\mathbf{S}_{u,v} \geq 0$ represents the influence of node u's label information on node v. This influence is nonzero only when $u \in \mathcal{N}(v) \cup v$, i.e., when u is a neighbor of v or v itself.

The label representation of each node is then computed iteratively over N steps. At each step $k \in \{1 \dots N\}$, the label representation of node v is updated by aggregating the label representations of its neighbors and itself, weighted by the influence scores in $\mathbf{S}$:

$$\mathbf{h}_{l_v}^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v) \cup v} \mathbf{S}_{u,v} \cdot \mathbf{h}_{l_u}^{(k-1)} \mathbf{W}^{(k)}\right), \tag{3.2}$$

where $\mathbf{W}$ is a learnable weight matrix and σ is a transformation function which is either an identity function or a non-linear transformation such as ReLU in our instantiation of our model. The following lemma (proof in Appendix) characterizes the label representations for the case when σ is an identity function.

Lemma 1. *Let $\mathcal{N}^{(N)}(v)$ denote the set of nodes that are reachable from node v in at most N steps, including v itself. When σ in (3.2) is instantiated as an identity function the final representation of v after N steps is given as*

$$\mathbf{h}_{l_v}^{(N)} = \left(\sum_{u \in \mathcal{N}^{(N)}(v)} \mathbf{S}_{u,v}^{N} \cdot \mathbf{y}_u\right) \mathbf{W}, \tag{3.3}$$

where $\mathbf{S}^N$ is the Nth power of the label influence matrix and $\mathbf{W} = \mathbf{W}^{(1)}\mathbf{W}^{(2)} \dots \mathbf{W}^{(N)}$.

A key advantage of our design is that application-specific prior knowledge about label influence can be directly encoded via the label influence matrix. Additionally, the one-hop label influence function can be parameterized and learned during training, allowing the model to adapt influence weights based on data. In our experiments we fixed $\mathbf{S}$ to be the symmetrically normalised adjacency matrix.

One potential concern in using the ground-truth labels of the training nodes $\mathcal{V}_l$ as input is that the model may simply memorizing and reuse the true labels for prediction without learning meaningful patterns. However, our model mitigates this risk by not keeping the

label representations of the training nodes fixed. Instead, these representations are iteratively updated during propagation, allowing them to incorporate information from their local neighborhood.

As a result, even training nodes receive influence from nearby nodes, enabling their label embeddings to reflect not only their original labels but also the label structure and correlations present in their N-hop neighborhoods. This dynamic updating encourages the model to learn shared patterns among neighboring nodes (rather than merely reinforcing known labels) and improves its ability to generalize to unseen nodes with diverse or partially overlapping label sets.

The Positional Representation Module. While the feature and label propagation modules are essential, each can be inadequate under certain conditions. For instance, in datasets like BLOGCAT, where node features are entirely absent, the feature representation module lacks the information needed to distinguish structurally similar nodes—limiting its expressive power. Conversely, the label representation module may struggle in settings with scarce labels. When few training nodes are available, large parts of the graph are excluded from influencing label propagation, since only labeled nodes contribute to the label representations. As a result, unlabeled nodes that could provide useful structural or semantic context are ignored, reducing the effectiveness of the label representation.

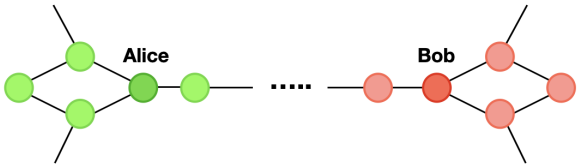
To address these limitations, we develop positional representations of all nodes in the graph based on only their relative positions in the graph. Unlike previous approaches that incorporate positional encodings as additional input feature dimensions and aggregate them locally [28], we argue that this aggregation can blur distinctions between nodes, similar to the smoothing of input features after feature aggregation, resulting in the loss of essential positional information. Although traditional methods such as Laplacian graph eigenvectors or more advanced techniques [21–23] can be employed as positional representations, the effectiveness of our approach comes from maintaining the positional representations as a separate module, preventing their aggregation within local neighborhoods. Our positional representation module is developed on two key intuitive assumptions:

- **Distance Limitation on Influence:** Nodes that are far apart in the graph are less likely to influence each other and, as a result, may belong to entirely different label sets. Our preliminary analysis of several datasets supports this assumption, indicating that it is sufficient to reconstruct the label set of a node by leveraging information from the label sets within its k-hop neighborhood for small k.

- **Influence-Driven Label Similarity:** The label similarity between two nodes u and v increases as the potential paths for influence propagation between them increase. In other words, the greater the opportunities for information or influence to flow between two nodes, the higher the likelihood that they will share similar labels.

To capture complementary patterns to what is already exploited by label representation module we learn node representations in an unsupervised manner without the actual information of the labels but under the above two assumptions. In general for a given pair of nodes u and v we are interested in generating low dimensional representations or embeddings while

Figure 3.5: Alice and Bob have similar local structures but have different interests (labels).



minimizing the following loss function

$$\mathscr{L}(u,v) = (1 - \mathscr{I}(u,v)) \cdot \mathrm{f}(\Phi_u, \Phi_v), \tag{3.4}$$

where $\mathscr{I}(u,v)$ quantifies the symmetrical influence of u and v on each other, $\mathrm{f}(.,.)$ is some function monotonically increasing in $\Phi_u \cdot \Phi_v$, where Φ_u and Φ_v denote the node representations of node u and v. Specifically for some small k we define valid node pairs to be at-most k-hop neighbors. For any valid pairs of nodes u and v we then quantify the influence $\mathscr{I}(u,v)$ as the probability of their co-occurrence in an infinite random walk over the underlying graph $\mathscr{G}$.

In our implementation, we use DEEPWALK to generate node positional encodings via short truncated random walks of length 10 and a window size of 5. These positional encodings are generated separately from the main model and are therefore used as fixed inputs to the module rather than being jointly learned with the rest of the framework. More generally, however, our framework is flexible, and the definitions of functions $\mathscr{I}(\cdot,\cdot)$ and $\mathrm{f}(\cdot,\cdot)$ can be adapted to suit the properties of a given dataset.

The Read-out Layer. Finally, we combine the representations from the three independent modules to input in a *Readout Layer* to get predictions for each node: $\mathbf{h}_v = \mathrm{COMB}(\mathbf{h}_{f_v}^{(K)}, \mathbf{h}_{l_v}^{(N)}, \Phi_v)$, where Φ_v corresponds to the embedding vector for node v generated by the node position representation module. The resulting representations are then used as input to the multi-label classifier (with output sigmoid layer as described in Section 3.2) which can be trained in an end-to-end manner using the binary cross entropy loss for the multi-label node classification task.

Analysing GNN-MULTIFIX's representations. To further support the design choices behind GNN-MULTIFIX, consider a simple social network graph $\mathscr{G}$, as illustrated in Figure 3.5, where two users, Alice and Bob, occupy structurally similar positions in their respective local neighborhoods but are located far apart in the graph. Despite their structural similarity, Alice and Bob are unlikely to share common interests or social connections due to the large graph distance between them. As a result, their labels (e.g., interests or affiliations) exert minimal influence on each other. An effective model should be able to distinguish between such nodes. In the absence of distinguishing node features, structurally similar nodes such as those representing Alice and Bob will be assigned similar feature representations due to their identical local structures. However, GNN-MULTIFIX can distinguish such nodes through two complementary mechanisms. In the first and simpler case, if the label distributions

in the N-hop neighborhoods of Alice and Bob are available and differ, then according to Equation (3.3), their label representations will also differ, allowing the model to distinguish them. In the second case, where all nodes in their N-hop neighborhoods are part of the test set and thus unlabeled, label propagation alone is insufficient. Here, GNN-MULTIFIX relies on its positional representation module, which captures global structural information. By optimizing the objective in Equation (3.4), the model learns positional embeddings that preserve graph-based distance influence $\mathscr{I}(u, v)$ between node pairs (u, v).

Our experiments on synthetic datasets further validate the effectiveness of GNN-MULTIFIX in challenging scenarios, including those with low-quality node features and low label homophily. Thanks to its modular design, GNN-MULTIFIX can adaptively leverage the most informative signal available—be it node features, label distributions, or positional information—allowing it to maintain strong performance even when certain inputs are weak or missing.

3.5. EXPERIMENTAL SET-UP

Datasets. For our experiments we employ 4 multi-label real-world graph datasets and 2 sets of synthetic datasets proposed in [1], where each set of the synthetic dataset consists of 5 datasets with varying feature quality and label homophily. We summarize the the characteristics of the datasets in Table 3.5 in Appendix 3.8.4. Below we provide a brief description of the datasets.

Real-world datasets We employ (i) BLOGCAT [5], in which nodes represent bloggers and edges their relationships, the labels denote the social groups a blogger is a part of, (ii)YELP [8], in which nodes correspond to the customer reviews and edges to their friendships with node labels representing the types of businesses (iii) DBLP [29], in which nodes represent authors and edges the co-authorship between the authors, and the labels indicate the research areas of the authors. and (iiii) PCG [1], in which proteins and the protein functional interactions are represented as nodes and edges. The labels of the nodes indicate the protein phenotypes of the corresponding proteins.

Synthetic datasets We obtain two types of synthetic datasets from [1]. Specifically we use (i) SYNTHETIC1 that consists of 5 graph datasets with the same graph structure in which the ratio of task-relevant and irrelevant feature varies in the range of $[0.0, 0.2, 0.5, 0.8, 1.0]$ and has label homophily of 0.2, (ii) SYNTHETIC2 that consists of 5 graph datasets in which the label homophily varies in the range of $[0.2, 0.4, 0.6, 0.8, 1.0]$. For the varying homophily experiment we only used the true node features.

Baselines. We compare our model against twelve baselines categorized from four different groups.

1. **Simple approaches**, which include Multilayer Perceptron (MLP [30]), which only uses node features and ignores the graph structure, DEEPWALK [31], which only uses graph structure and ignores the node features, and MAJORITYVOTE, which takes the direct neighbors of the test nodes which are in the training set and sum the votes of their labels.

The votes are then normalized by the total number of votes in the direct neighborhood to get probabilities.

2. **Classical GNNs and variants** including (a) GCN [2], GAT [3], and GRAPHSAGE [4], which are known to perform well for graphs with high label homophily, and (b)H2GCN [32] and FSGNN [33], which are designed to perform well both on homophilic and heterophilic graphs by separating the ego embedding and aggregated embeddings from different orders of neighborhoods. The difference is FSGNN learns attention scores to important features from different hops neighborhoods.

3. **Label informed GNNs**, which leverage label information of training nodes beyong the loss function to extract more informative node representations. From this category, we include (a) GCN-LPA [10], which combines label propagation and GCN for node classification, and (b) LANC [11], which employ convolutional operations to extract representations from node's local neighborhoods and merge them with label embeddings for the final classification and is specifically designed for multi-label node classification.

4. **GNN with structural and positional representations**, which integrates the positional information of the node as part of the input to obtain highly expressive models, we include GNN-LSPE [28] and ID-GNN-FAST [24] from this category, which augment the input features using the positional representations and injecting the identities of the nodes, respectively.

Instantiations of GNN-MULTIFIX. Let $\tilde{\mathbf{D}}$ denotes the degree matrix of the graph with added self loops i.e. with adjacency $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. For any layer k, we instantiate the aggregation function in the feature representation module as $\mathbf{Z}^{(k)} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(k)}$ and the combination function as $\mathbf{H}^{(k+1)} = \sigma\left(\mathbf{Z}^{(k)}\mathbf{W}^{(l)}\right)$, where the σ is either an identity function or a ReLU. For label representation module we set $\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$. In the readout layer we concatenate the three reprsentations and feed it to a multi-label classifier. As variations of our model, we develop (i) GNN-MULTIFIX-LINEAR, where the σ functions in feature and label representation modules is an identity function and the classifier is implemented as a single linear layer. (ii) GNN-MULTIFIX-MLP1, where σ in the representation modules corresponds to ReLU and one-layer MLP is used as a classifier. (iii) we use GNN-MULTIFIX-MLP3 with three layer MLP for further improvements on the node classification task. σ in both modules corresponds to ReLU. The explicit hyperparameter settings of all the baselines as well as GNN-MULTIFIX can be found in Appendix 3.8.4 and 3.8.4.

Evaluation. Motivated by [1, 34, 35], we compute the **Average Precision (AP) score** across three random splits of the collected datasets, with the exception of PCG, for which we use the predefined split as in [1]. In random splits, we allocate 60% of nodes for training, 20% for validation, and 20% for testing purposes.

3.6. RESULTS AND DISCUSSION

We discuss the large-scale experiment results conducted on the 4 real-world datasets in Section 3.6.1 and 2 sets of synthetic datasets in Section 3.6.2. Due to the space limit, we

provide the hyperparameter settings and detailed performance scores with standard deviation in appendix 3.8.4. We also present an ablation study of ablation study of GNN-MULTIFIX in Appendix 3.8.3.

3.6.1. RESULTS ON REAL-WORLD DATASETS

In Table 3.1, we summarize the results on four real-world datasets. **Firstly,** it is surprising that the simple MAJORITYVOTE baseline outperforms most of the baselines methods for BLOGCAT. It further supports our hypothesis that existing GNN methods without appropriate feature information fail to effectively distinguish nodes with differing labels. Even when incorporating additional label information into the input, models like GCN-LPA fail to demonstrate improvement. Methods augmenting input features with positional encoding, such as GNN-LSPE and ID-GNN-FAST, fail to surpass the performance of DEEPWALK. Notably, the simplest version of GNN-MULTIFIX-LINEAR yields improvements of 18.4% compared to the best baseline model, which underscores the effectiveness of directly integrating label information into the input and employing positional representation to distinguish nodes with differing labels.

Secondly, in datasets with a high number of labels per node, such as YELP, low homophily arises from diverse label assignments, however, nodes that share the same subset of their labels with the ego node are in the local neighborhoods. Consequently, straightforward baselines like MAJORITYVOTE outperform more complex methods such as MLP, DEEPWALK, and GCN-LPA. Interestingly, GNNs combined with sampling techniques in the local neighborhood, such as GRAPHSAGE, demonstrate superior performance compared to other designs of GNN. Nevertheless, our model (which uses GCN as base feature representation module) performs comparably well with the winning model in these scenarios. Performance of GNN-MULTIFIX can be further enhanced by instantiating feature representation module by GRAPHSAGE's message passing operation.

Thirdly for DBLP which show very high label homophily, various GNN designs perform comparably well, surpassing simple baselines like MLP and DEEPWALK. By incorporating additional positional representation and label representation, our model outperform all the other baselines.

Finally, for PCG which has low label homophily and high clustering coefficient, baselines using random walk to explore the diverse local neighborhood such as GNN-LSPE and DEEPWALK outperform MLP, and more sophisticated CNN- and GNN-based methods. The competitive performance of MAJORITYVOTE despite PCG's low label homophily indicates that the nodes which share the similar labels with the ego node lies in the local neighborhood, which further explains the low performances of H2GCN and FSGNN which rely on leveraging higher order neighborhood for low-homophily multi-class datasets.

3.6.2. RESULTS ON SYNTHETIC DATASETS

To compare the performance of various methods under varying feature quality and label homophily, we experimented with two synthetic datasets. Results are provided in 3.2. Our GNN-MULTIFIX outperforms all compared models baselines for most of the cases.

Varying Feature Quality. Across all levels of feature quality, the performance of DEEPWALK remains consistent, as it generates node representations solely based on the graph structure. The MLP is predictably the most sensitive method to changes in feature quality,

Table 3.1: Mean performance scores (Average Precision) on real-world datasets. The best score is marked in bold. The second best score is marked with underline. "OOM" denotes the "Out Of Memory" error.

Method	BLOGCAT	YELP	DBLP	PCG
MLP	0.043	0.096	0.350	0.148
DEEPWALK	0.190	0.096	0.585	0.229
MAJORITYVOTE	0.133	0.112	0.869	0.203
LANC	0.050	OOM	0.836	0.185
GCN	0.037	0.131	0.893	0.210
GAT	0.041	0.150	0.829	0.168
GRAPHSAGE	0.045	**0.251**	0.868	0.185
H2GCN	0.039	0.226	0.858	0.192
FSGNN	0.075	0.093	0.858	0.163
GCN-LPA	0.043	0.116	0.801	0.167
GNN-LSPE	0.165	OOM	0.590	**0.528**
ID-GNN-FAST	0.037	0.134	0.857	0.235
GNN-MULTIFIX-LINEAR	**0.225**	0.200	<u>0.935</u>	0.242
GNN-MULTIFIX-MLP1	<u>0.220</u>	0.217	0.931	<u>0.254</u>
GNN-MULTIFIX-MLP3	0.127	<u>0.238</u>	**0.942**	0.252

as it completely disregards the graph structure. LANC also demonstrates sensitivity to variations in feature quality, extracting feature vectors from the local neighborhood through convolutional operations on the stacked feature matrix of direct neighbors. Among the classical GNNs and their variations, GRAPHSAGE and H2GCN exhibit the most significant improvements with increasing feature quality. With the integration of label propagation and positional encoding, GCN-LPA and ID-GNN-FAST show limited but stable performance across varying feature qualities. In contrast, our model and its variants achieve superior and robust performance across different levels of feature quality, with the most notable improvement observed at the lowest feature quality. Notably, in the synthetic datasets where the input features are strongly correlated with the labels [1], MLP outperforms all other baselines at the highest feature quality level.

Varying Label Homophily. The performance of the MLP remains relatively consistent across different levels of label homophily, primarily because it relies solely on input features. Conversely, while considerable effort has been invested in developing complex methods for node classification tasks, we find that simpler baselines like DEEPWALK surpass standard GNNs in several instances. Specifically, on synthetic datasets with label homophily levels of 0.4 and 0.6, DEEPWALK emerges as the top-performing method. Given that LANC extracts feature vectors from the local neighborhood through convolutional operations on the stacked feature matrix of direct neighbors, it is not surprising that its performance is highly sensitive to changes in label homophily. All classical GNNs and their variants exhibit similar sensitivity to variations in label homophily, demonstrating improved performance under high label homophily conditions. Although GCN-LPA and ID-GNN-FAST integrate label information to adjust edge weights and enhance input features with positional encoding,

Table 3.2: Average Precision (mean) on the synthetic datasets with varying levels of feature quality and label homophily (see definition 4). r_{ori_feat} and r_{homo} refer to the fraction of original features and the varying label homophily of the graph with all true features, respectively. It was not possible to run GNN-LSPE on our synthetic datasets due to its large runtime.

Method	r_{ori_feat}					r_{homo}				
	0.0	0.2	0.5	0.8	1.0	0.2	0.4	0.6	0.8	1.0
MLP	0.171	0.185	0.222	0.266	**0.336**	**0.362**	0.354	0.355	0.360	0.355
DEEPWALK	0.181	0.181	0.181	0.181	0.181	0.181	0.522	**0.813**	0.869	0.552
LANC	0.191	0.191	0.192	0.211	0.215	0.179	0.397	0.473	0.500	0.634
GCN	0.261	0.261	0.261	0.262	0.262	0.261	0.308	0.360	0.430	0.450
GAT	0.171	0.169	0.176	0.177	0.182	0.196	0.341	0.353	0.399	0.403
GRAPHSAGE	0.171	0.191	0.232	0.268	0.304	0.317	0.408	0.437	0.484	0.491
H2GCN	0.182	0.202	0.224	0.258	0.290	0.300	0.487	0.519	0.617	0.618
FSGNN	0.180	0.191	0.211	0.228	0.243	0.240	0.433	0.419	0.328	0.371
GCN-LPA	0.174	0.175	0.170	0.170	0.172	0.174	0.277	0.352	0.414	0.346
ID-GNN-FAST	0.261	0.261	0.261	0.261	0.261	0.261	0.308	0.363	0.432	0.457
GNN-MULTIFIX-LINEAR	0.249	0.248	0.248	0.248	0.247	0.256	0.468	0.687	0.812	0.445
GNN-MULTIFIX-MLP1	**0.289**	**0.283**	**0.283**	**0.280**	0.288	0.288	0.492	0.701	0.910	**0.832**
GNN-MULTIFIX-MLP3	0.270	0.261	0.255	0.254	0.267	0.269	**0.527**	0.795	**0.916**	0.811

their improvements over simple classical GNNs are limited. In contrast, our model and its variants demonstrate significant enhancements. Notably, at the lowest level of homophily, where neighbors are quite dissimilar compared to ego nodes and input features are strongly correlated with the labels, the MLP outperforms all other baselines.

Performance of majority vote. In Table 3.3 we provide performance of MAJORITYVOTE baseline for our synthetic datasets for varying label homophily. Unlike for real world datasets majority vote baseline outperforms all methods. For low homophilic synthetic graphs we attribute this finding to high edge density of the synthetic graphs when the homophily is low. Note that the edge density is close to 0.27 for SYNTHETIC1 which has homophily level of 0.2. This turns out to be on average around 800 1-hop neighbors per node when the maximum number of labels per node is 12.

While the edge density decreases with increasing homophily, the performance of MAJORITYVOTE still stands out to the best. To understand this phenomena consider the extreme case of homophily 1. On average each has 15 neighbors but all of its neighbors have the same label set as the node itself. The problem then becomes trivial for MAJORITYVOTE baseline which can give the correct answer even if only one node in the neighborhood is labeled. Interestingly while MAJORITYVOTE is supposedly a weak baseline in real world datasets it provides an upper bound of performance on chosen synthetic datasets. Our method nevertheless outperforms other baselines in most of the settings. For instance, at the highest homophily level of 1 for SYNTHETIC2 dataset we achieve average precision (0.832) comparable to the MAJORITYVOTE (0.847) and an improvement of 31.23% over the second best method (LANC).

Table 3.3: Average Precision (mean) on the synthetic datasets with varying levels label homophily when MAJORITYVOTE is employed.

Method	r_{homo_clean}				
	0.2	0.4	0.6	0.8	1.0
MAJORITYVOTE	0.477	0.926	0.957	0.938	0.847

3.7. CONCLUSION

We delve into the often-overlooked scenario of multi-label node classification, revealing that GNNs struggle to learn from multi-label datasets, even with abundant training data. We show that even the most expressive (in terms of distinguishing between non-isomorphic graphs) GNNs may fail to effectively distinguish nodes with different label-set without node attributes and explicit label information. Our simple yet effective approach, GNN-MULTIFIX, which leverages feature, label, and positional information of a node to predict its labels consistently outperforms existing methods on multi-label node classification task.

3.8. APPENDIX

Organization of the Appendix. In Section 3.8.1, we provide pseudo code for the algorithm of GNN-MULTIFIX. We also include a theoretical analysis of GNN-MULTIFIX in section 4.3.3. In section 3.8.3, we provide ablation studies on the submodules in GNN-MULTIFIX. In the following Section 3.8.4, we provide supplementary materials about the experiments conducted in this work. Specifically, in 3.8.4, we summarize the detailed characteristics of the datasets used in this work. Furthermore, The hyperparameter setting to reproduce the results in this work are summarized in Section 3.8.4 and 3.8.4. We provide the complete results with standard deviation on the three random splits described in the main paper in section 3.8.4.

3.8.1. PSEUDOCODE FOR GNN-MULTIFIX

In this section, we summarize the general framework for the node classification task on graph-structured data, which we refer to as GNN-MULTIFIX. As shown in Algorithm 1 from line 2 to 5 and 6 to 9, for each node $v \in \mathcal{V}$, we first perform K layers of feature propagation and N layer of label propagation to generate feature representation and label representation, respectively. Note that any aggregation and message-passing functions from classical GNNs can be used as AGG and COMB, and they are not restricted to be identical for feature and label propagation, which makes our method a general framework for the node classification task, as any other GNNs can be used as backbone model. In the implementation used in the experiment Section 5.4, we use the operation AGG and COMB from GCN.

Then, we generate N_{rm} number of random walks and train a $SkipGram$ model to obtain the positional embedding ϕ_v. In this stage, the embeddings of the nodes that are far away from each other in the graph are forced to be different.

In line 11, we COMB the representations generated from input feature, input label, and the positional information to obtain the overall representation for node v, denoted as $\mathbf{h}_v$.

Finally, we feed $\mathbf{h}_v$ into a ReadOut layer, which is typically a MLP with *Sigmoid*, to obtain the predicted probabilities.

3.8.2. MISSING PROOF

Proof of Lemma 1. We can rewrite (3.2) in the matrix form as $\mathbf{H}^{(k)} = \sigma(\mathbf{S}\mathbf{H}^{(k-1)}\mathbf{W}^{(k)})$ where $\mathbf{H}^{(k)}$ is the corresponding label representation matrix at step k. When σ is identity we can write the final representation matrix after N steps as

$$\begin{aligned}\mathbf{H}^{(N)} &= \mathbf{S}\mathbf{H}^{(N-1)}\mathbf{W}^{(N)} = \mathbf{S}^2\mathbf{H}^{(N-2)}\mathbf{W}^{(N-1)}\mathbf{W}^{(N)} \\ &= \mathbf{S}^3\mathbf{H}^{(N-3)}\mathbf{W}^{(N-2)}\mathbf{W}^{(N-1)}\mathbf{W}^{(N)} = \ldots = \mathbf{S}^N\mathbf{H}^0\mathbf{W},\end{aligned} \tag{3.5}$$

where $\mathbf{W} = \mathbf{W}^{(1)}\mathbf{W}^{(2)}\ldots\mathbf{W}^{(N)}$. Equation (3.3) corresponds to single row in $\mathbf{H}^{(N)}$. □

3.8.3. ABLATION STUDY

In this section, we verify the effectiveness of our design by disabling one module at a time in GNN-MULTIFIX-LINEAR. There are three module in GNN-MULTIFIX, the feature representation module, the label representation module, and the positional encoding module. Thus, we generate three variations of GNN-MULTIFIX-LINEAR, i.e., GNN-MULTIFIX-LINEAR without feature representation module, denoted as GNN-MULTIFIX-LINEAR w/o

Algorithm 1 Learning node representations with GNN-MULTIFIX

Require: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; input node features $\mathbf{X}$; padded label matrix $\mathbf{h}_{l_v}^{(0)}$; number of feature propagation K; number of label propagation N; Aggregation function AGG(.); Combine function COMB(.); Read-out layer ReadOut(.)

Ensure: Prediction matrix $\hat{\mathbf{Y}}_v \in \mathbb{R}^{n \times C}$

for $v \in \mathcal{V}$ **do**
  Initialize input features $\mathbf{h}_{f_v}^{(0)} = \mathbf{X}$
  for $i \in K$ **do**
    $\mathbf{h}_{f_{v\_\mathcal{N}(v)}}^{(i)} = \text{AGG}_{(f)}^{(i)}\left(\left\{\mathbf{h}_{f_u}^{(i-1)} : u \in \mathcal{N}(v)\right\}\right)$
    $\mathbf{h}_{f_v}^{(i)} = \text{COMB}_{(f)}^{(i)}\left(\mathbf{h}_{f_{v\_\mathcal{N}(v)}}^{(i)}, \mathbf{h}_{f_v}^{(i-1)}\right)$
  Initialize input label matrix $\mathbf{h}_{l_v}^{(0)}$
  for $j \in N$ **do**
    $\mathbf{h}_{l_{v\_\mathcal{N}(v)}}^{(j)} = \text{AGG}_{(l)}^{(j)}\left(\left\{\mathbf{h}_{l_u}^{(j-1)} : u \in \mathcal{N}(v)\right\}\right)$
    $\mathbf{h}_{\mathbf{l_v}}^{(\mathbf{j})} = \sigma_{(l)}^{(N)}\left(\mathbf{h}_{l_{v\_\mathcal{N}(v)}}^{(j)}\right)$
  Generate positional embeddings Φ_v
  Generate input representation for node v: $\mathbf{h}_v = \text{COMB}(\mathbf{h}_{f_v}^{(K)}, \mathbf{h}_{l_v}^{(N)}, \Phi_v)$
  $\hat{\mathbf{Y}}_{\mathbf{v}} = \text{ReadOut}(\mathbf{h}_v)$
$\hat{\mathbf{Y}} := [\hat{\mathbf{Y}}_{v_1}, \hat{\mathbf{Y}}_{v_2}, \ldots, \hat{\mathbf{Y}}_{v_n}]$

Table 3.4: Ablation study on GNN-MULTIFIX-LINEAR by disabling each module at a time on the dataset BLOGCAT and DBLP.

	BLOGCAT	DBLP
GNN-MULTIFIX-LINEAR w/o FR	0.225	0.920
GNN-MULTIFIX-LINEAR w/o LR	0.225	0.864
GNN-MULTIFIX-LINEAR w/o PE	0.045	0.934
GNN-MULTIFIX-LINEAR	0.225	0.935

FR, GNN-MULTIFIX-LINEAR without label representation module, denoted as GNN-MULTIFIX-LINEAR w/o LR, and GNN-MULTIFIX-LINEAR without positional encoding module, denoted as GNN-MULTIFIX-LINEAR w/o PE and test the performances of them on the dataset BLOGCAT and DBLP and summarize the results in the table 3.4.

BLOGCAT is a dataset without input features and exhibits the lowest label homophily. As a result, disabling the feature and label representation modules has little effect on the final performance. However, disabling the positional encoding module leads to a significant performance drop. This indicates that when input features are poor, and the labels in the local neighborhood provide little useful information for inferring the labels of a node, GNN-MULTIFIX adapts by relying more on the positional encoding module. In contrast, DBLP, a dataset with the highest label homophily, shows that input features are limited in their ability to infer node labels, as seen in the weaker performance of MLP on this dataset. Here, disabling the label propagation module causes the largest performance drop. By comparing

these results on BLOGCAT and DBLP with differing characteristics, we demonstrate that our model effectively identifies and leverages the most informative aspects of the input to predict node labels.

3.8.4. SUPPLEMENTARY MATERIAL FOR EXPERIMENTS

DESCRIPTION OF DATASETS

The detailed statistics of the datasets are provided in Table 3.5.

Table 3.5: $|\mathcal{V}|$, $|\mathcal{E}|$, $|\mathcal{F}|$ denote the number of nodes, edges and the node feature dimension respectively. $clus$ and r_{homo} denote the clustering coefficient and the label homophily (average of Jaccard similarity of labels over all edges). C is the total number of labels. ℓ_{med}, ℓ_{mean}, and ℓ_{max} specify the median, mean, and max values corresponding to the number of labels of a node. '25%', '50%', and '75%' corresponds to the 25th, 50th, and 75th percentiles of the sorted list of the number of labels for a node. "N.A." and "vary" indicate the corresponding characteristic is not available or varying in the graphs.

DATASET	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{F}\rvert$	$clus$	r_{homo}	C	ℓ_{med}	ℓ_{mean}	ℓ_{max}	25%	50%	75%
BLOGCAT	$10K$	$333K$	*N.A.*	0.46	0.10	39	1	1.40	11	1	1	2
YELP	$716K$	$7.34M$	300	0.09	0.22	100	6	9.44	97	3	6	11
DBLP	$28K$	$68K$	300	0.61	0.76	4	1	1.18	4	1	1	1
PCG	$3K$	$37K$	32	0.34	0.17	15	1	1.93	12	1	1	2
SYNTHETIC1	$3K$	$2.37M$	vary	0.53	0.2	20	3	3.23	12	1	3	5
SYNTHETIC2	$3K$	vary	10	vary	vary	20	3	3.23	12	1	3	5

HYPERPARAMETER SETTING FOR BASELINES

For a fair comparison, we also tuned the hyperparameters of the collected baselines and summarized the optimal settings for each dataset. Specifically, we tuned the learning rate parameters in $\{0.1, 0.01, 0.001, 0.0001\}$ and the hidden units for neural networks in $\{64, 128, 256, 512\}$. Additionally, we adopted a common deployment of neural networks with 2 layers and used early stopping with a patience of 100 for all methods in this work, along with a weight decay of $5e^{-4}$. The specific parameters of the collected baseline models are fixed to the setting in the original paper. We release all the detailed hyperparameter setting in https://anonymous.4open.science/r/Graph-MultiFix-4121.

HYPERPARAMETER SETTING FOR GNN-MULTIFIX

We summarize the hyperparameter setting used in the variants of GNN-MULTIFIX for reproducing the experimental results in this work according to each dataset in Table 3.6. We use the consistant notations as in the main paper. k represents the number of feature propagation, N indicates the number of label propagation. We also used early stopping with the patience specified in the table to prevent the models from overfitting. The hidden dimension are used for both feature representation module and label representation module. And Φ_{dim} indicates the dimension of positional representations.

Table 3.6: The hyperparameter setting for variations of GNN-MULTIFIX. K and N indicate the number of layers used in the feature and label propagation module, respectively.

		K	N	patience	learning rate	hidden dim	Φ_{dim}
GNN-MULTIFIX-LINEAR	BLOGCAT	2	1	100	0.03	256	64
	YELP	2	25	100	0.005	256	64
	DBLP	2	2	100	0.01	256	64
	PCG	2	1	100	0.01	256	64
GNN-MULTIFIX-MLP1	BLOGCAT	2	1	100	0.01	256	64
	YELP	2	25	100	0.005	256	64
	DBLP	2	2	100	0.01	256	64
	PCG	2	5	100	0.01	256	64
GNN-MULTIFIX-MLP3	BLOGCAT	2	1	100	0.01	256	64
	YELP	2	25	100	0.005	256	64
	DBLP	2	2	100	0.01	256	64
	PCG	2	5	100	0.01	256	64

COMPLETE EXPERIMENT RESULTS

Due to the page limit, we provide the full experiment results with standard deviation of three random splits on the collected real-world datasets and the synthetic datasets in the following Table 3.7, 3.8 and 3.9, respectively.

Table 3.7: Mean performance scores (Average Precision) on real-world datasets. The best score is marked in bold. The second best score is marked with underline. "OOM" denotes the "Out Of Memory" error.

Method	BLOGCAT	YELP	DBLP	PCG
MLP	0.043 ± 0.006	0.096 ± 0.000	0.350 ± 0.001	0.148 ± 0.006
DEEPWALK	0.190 ± 0.002	0.096 ± 0.000	0.585 ± 0.003	0.229 ± 0.010
MAJORITYVOTE	0.133 ± 0.000	0.112 ± 0.000	0.869 ± 0.000	0.203 ± 0.000
LANC	0.050 ± 0.001	OOM	0.836 ± 0.010	0.185 ± 0.011
GCN	0.037 ± 0.000	0.131 ± 0.001	0.893 ± 0.002	0.210 ± 0.003
GAT	0.041 ± 0.001	0.150 ± 0.001	0.829 ± 0.002	0.168 ± 0.022
GRAPHSAGE	0.045 ± 0.001	$\mathbf{0.251 \pm 0.003}$	0.868 ± 0.002	0.185 ± 0.003
H2GCN	0.039 ± 0.001	0.226 ± 0.005	0.858 ± 0.006	0.192 ± 0.005
FSGNN	0.075 ± 0.000	0.093 ± 0.000	0.858 ± 0.000	0.163 ± 0.000
GCN-LPA	0.043 ± 0.003	0.116 ± 0.007	0.801 ± 0.012	0.167 ± 0.001
GNN-LSPE	0.165 ± 0.000	OOM	0.590 ± 0.000	$\mathbf{0.528 \pm 0.005}$
ID-GNN-FAST	0.037 ± 0.000	0.134 ± 0.000	0.857 ± 0.000	0.235 ± 0.010
GNN-MULTIFIX-LINEAR	$\mathbf{0.225 \pm 0.010}$	0.201 ± 0.010	$\underline{0.935 \pm 0.000}$	0.242 ± 0.010
GNN-MULTIFIX-MLP1	$\underline{0.220 \pm 0.000}$	0.217 ± 0.000	0.931 ± 0.000	$\underline{0.254 \pm 0.010}$
GNN-MULTIFIX-MLP3	0.127 ± 0.000	$\underline{0.237 \pm 0.000}$	$\mathbf{0.942 \pm 0.010}$	0.252 ± 0.010

Table 3.8: Average Precision (mean) with standard deviation on the synthetic datasets with varying levels of feature quality. r_{ori_feat} refers to the fraction of original features of the graph with all true features. It was not possible to run GNN-LSPE on our synthetic datasets due to its large runtime.

Method	r_{ori_feat} 0.0	0.2	0.5	0.8	1.0
MLP	0.171 ± 0.010	0.185 ± 0.000	0.222 ± 0.000	0.266 ± 0.010	0.336 ± 0.000
DEEPWALK	0.181 ± 0.007	0.181 ± 0.007	0.181 ± 0.007	0.181 ± 0.007	0.181 ± 0.007
MAJORITYVOTE	$\mathbf{0.477 \pm 0.020}$	$\mathbf{0.477 \pm 0.020}$	$\mathbf{0.477 \pm 0.020}$	$\mathbf{0.477 \pm 0.020}$	$\mathbf{0.477 \pm 0.020}$
LANC	0.191 ± 0.020	0.191 ± 0.020	0.192 ± 0.010	0.211 ± 0.020	0.215 ± 0.020
GCN	0.261 ± 0.010	0.261 ± 0.010	0.261 ± 0.010	0.262 ± 0.010	0.262 ± 0.010
GAT	0.171 ± 0.000	0.169 ± 0.000	0.176 ± 0.000	0.177 ± 0.000	0.182 ± 0.010
GRAPHSAGE	0.171 ± 0.000	0.191 ± 0.000	0.232 ± 0.010	0.268 ± 0.000	$\underline{0.304 \pm 0.010}$
H2GCN	0.182 ± 0.000	0.202 ± 0.010	0.224 ± 0.000	0.258 ± 0.010	0.290 ± 0.010
FSGNN	0.180 ± 0.000	0.191 ± 0.000	0.211 ± 0.000	0.228 ± 0.010	0.243 ± 0.000
GCN-LPA	0.174 ± 0.000	0.175 ± 0.000	0.170 ± 0.000	0.170 ± 0.000	0.172 ± 0.010
ID-GNN-FAST	0.261 ± 0.010	0.261 ± 0.010	0.261 ± 0.010	0.261 ± 0.010	0.261 ± 0.010
GNN-MULTIFIX-LINEAR	0.249 ± 0.010	0.248 ± 0.010	0.248 ± 0.010	0.248 ± 0.010	0.247 ± 0.010
GNN-MULTIFIX-MLP1	$\underline{0.289 \pm 0.010}$	$\underline{0.283 \pm 0.020}$	$\underline{0.283 \pm 0.010}$	$\underline{0.280 \pm 0.010}$	0.288 ± 0.010
GNN-MULTIFIX-MLP3	0.270 ± 0.020	0.261 ± 0.010	0.255 ± 0.010	0.254 ± 0.010	0.267 ± 0.020

Table 3.9: Average Precision (mean) with standard deviation on the synthetic datasets with varying levels of label homophily (see definition 4). r_{homo} refers to the fraction of the varying label homophily of the graph with all true features. It was not possible to run GNN-LSPE on our synthetic datasets due to its large runtime.

Method	r_{homo_clean} 0.2	0.4	0.6	0.8	1.0
MLP	$\underline{0.362 \pm 0.000}$	0.354 ± 0.000	0.355 ± 0.010	0.360 ± 0.010	0.355 ± 0.010
DEEPWALK	0.181 ± 0.007	0.522 ± 0.014	$\underline{0.813 \pm 0.010}$	0.869 ± 0.019	0.552 ± 0.007
MAJORITYVOTE	$\mathbf{0.477 \pm 0.020}$	$\mathbf{0.926 \pm 0.010}$	$\mathbf{0.957 \pm 0.010}$	$\mathbf{0.938 \pm 0.010}$	$\mathbf{0.847 \pm 0.020}$
LANC	0.179 ± 0.010	0.397 ± 0.020	0.473 ± 0.030	0.500 ± 0.030	0.634 ± 0.050
GCN	0.261 ± 0.010	0.308 ± 0.010	0.360 ± 0.000	0.430 ± 0.010	0.450 ± 0.010
GAT	0.196 ± 0.010	0.341 ± 0.010	0.353 ± 0.000	0.399 ± 0.020	0.403 ± 0.010
GRAPHSAGE	0.317 ± 0.010	0.408 ± 0.000	0.437 ± 0.000	0.484 ± 0.000	0.491 ± 0.010
H2GCN	0.300 ± 0.010	0.487 ± 0.010	0.519 ± 0.010	0.617 ± 0.030	0.618 ± 0.010
FSGNN	0.240 ± 0.000	0.433 ± 0.030	0.419 ± 0.010	0.328 ± 0.070	0.371 ± 0.080
GCN-LPA	0.174 ± 0.010	0.277 ± 0.000	0.352 ± 0.010	0.414 ± 0.000	0.346 ± 0.010
ID-GNN-FAST	0.261 ± 0.010	0.308 ± 0.010	0.363 ± 0.000	0.432 ± 0.010	0.457 ± 0.010
GNN-MULTIFIX-LINEAR	0.256 ± 0.010	0.468 ± 0.010	0.687 ± 0.030	0.812 ± 0.020	0.445 ± 0.020
GNN-MULTIFIX-MLP1	0.288 ± 0.010	0.492 ± 0.010	0.701 ± 0.000	0.910 ± 0.020	$\underline{0.832 \pm 0.030}$
GNN-MULTIFIX-MLP3	0.269 ± 0.030	$\underline{0.527 \pm 0.010}$	0.795 ± 0.020	$\underline{0.916 \pm 0.010}$	0.811 ± 0.040

REFERENCES

1. Zhao, T., Dong, T. N., Hanjalic, A. & Khosla, M. Multi-label Node Classification On Graph-Structured Data. *Transactions on Machine Learning Research.* ISSN: 2835-8856. https://openreview.net/forum?id=EZhkV2BjDP (2023).
2. Kipf, T. N. & Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907* (2016).
3. Veličković, P. *et al.* Graph Attention Networks. *International Conference on Learning Representations.* accepted as poster. https://openreview.net/forum?id=rJXMpikCZ (2018).
4. Hamilton, W. L., Ying, R. & Leskovec, J. Inductive Representation Learning on Large Graphs. *CoRR* **abs/1706.02216.** arXiv: 1706.02216. http://arxiv.org/abs/1706.02216 (2017).
5. Tang, L. & Liu, H. *Relational Learning via Latent Social Dimensions* in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, Paris, France, 2009), 817–826. ISBN: 978-1-60558-495-9. https://doi.org/10.1145/1557019.1557109.
6. Zheng, X. *et al.* Towards Data-centric Graph Machine Learning: Review and Outlook. *arXiv preprint arXiv:2309.10979* (2023).
7. Srinivasan, B. & Ribeiro, B. On the equivalence between positional node embeddings and structural graph representations. *arXiv preprint arXiv:1910.00452* (2019).
8. Zeng, H., Zhou, H., Srivastava, A., Kannan, R. & Prasanna, V. K. GraphSAINT: Graph Sampling Based Inductive Learning Method. *CoRR* **abs/1907.04931.** arXiv: 1907.04931. http://arxiv.org/abs/1907.04931 (2019).
9. Wang, B., Tu, Z. & Tsotsos, J. K. *Dynamic Label Propagation for Semi-supervised Multi-class Multi-label Classification* in *2013 IEEE International Conference on Computer Vision* (2013), 425–432.
10. Wang, H. & Leskovec, J. Unifying Graph Convolutional Neural Networks and Label Propagation. *CoRR* **abs/2002.06755.** arXiv: 2002.06755. https://arxiv.org/abs/2002.06755 (2020).
11. Zhou, C. *et al.* Multi-label graph node classification with label attentive neighborhood convolution. *Expert Systems with Applications* **180,** 115063. ISSN: 0957-4174. https://www.sciencedirect.com/science/article/pii/S0957417421005042 (2021).
12. Sato, R. *Training-free Graph Neural Networks and the Power of Labels as Features* 2024. arXiv: 2404.19288 [cs.LG].
13. Bellei, C., Alattas, H. & Kaaniche, N. Label-GCN: An Effective Method for Adding Label Propagation to Graph Convolutional Networks. *CoRR* **abs/2104.02153.** arXiv: 2104.02153. https://arxiv.org/abs/2104.02153 (2021).
14. Sun, Y. *et al. Multi-Label Node Classification with Label Influence Propagation* in *The Thirteenth International Conference on Learning Representations* (2025). https://openreview.net/forum?id=3X3LuwzZrl.

15. Chen, Z.-M., Wei, X.-S., Wang, P. & Guo, Y. *Multi-label image recognition with graph convolutional networks* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), 5177–5186.

16. Huang, S.-J. & Zhou, Z.-H. Multi-Label Learning by Exploiting Label Correlations Locally. *Proceedings of the AAAI Conference on Artificial Intelligence* **26,** 949–955. https://ojs.aaai.org/index.php/AAAI/article/view/8287 (Sept. 2021).

17. Lanchantin, J., Sekhon, A. & Qi, Y. *Neural message passing for multi-label classification* in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2019), 138–163.

18. Saini, D. *et al. GalaXC: Graph Neural Networks with Labelwise Attention for Extreme Classification* in (Association for Computing Machinery, Ljubljana, Slovenia, 2021). ISBN: 978-1-4503-8312-7. https://doi.org/10.1145/3442381.3449937.

19. Shi, M., Tang, Y., Zhu, X. & Liu, J. Multi-Label Graph Convolutional Network Representation Learning. *CoRR* **abs/1912.11757.** arXiv: 1912.11757. http://arxiv.org/abs/1912.11757 (2019).

20. Zhu, Y., Kwok, J. T. & Zhou, Z. Multi-Label Learning with Global and Local Label Correlation. *CoRR* **abs/1704.01415.** arXiv: 1704.01415 (2017).

21. Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y. & Bresson, X. Graph Neural Networks with Learnable Structural and Positional Representations. *CoRR* **abs/2110.07875.** arXiv: 2110.07875. https://arxiv.org/abs/2110.07875 (2021).

22. Eliasof, M. *et al. Graph Positional Encoding via Random Feature Propagation* in *Proceedings of the 40th International Conference on Machine Learning* (eds Krause, A. *et al.*) **202** (PMLR, 23–29 Jul 2023), 9202–9223. https://proceedings.mlr.press/v202/eliasof23a.html.

23. Park, W., Chang, W., Lee, D. & Kim, J. Graph Self-Attention for learning graph representation with Transformer. *CoRR* **abs/2201.12787.** arXiv: 2201.12787. https://arxiv.org/abs/2201.12787 (2022).

24. You, J., Gomes-Selman, J., Ying, R. & Leskovec, J. Identity-aware Graph Neural Networks. *CoRR* **abs/2101.10320.** arXiv: 2101.10320. https://arxiv.org/abs/2101.10320 (2021).

25. Agarwal, C. & Hooker, S. Estimating Example Difficulty using Variance of Gradients. *CoRR* **abs/2008.11600.** arXiv: 2008.11600. https://arxiv.org/abs/2008.11600 (2020).

26. Siddiqui, S. A., Rajkumar, N., Maharaj, T., Krueger, D. & Hooker, S. *Metadata Archaeology: Unearthing Data Subsets by Leveraging Training Dynamics* 2022. arXiv: 2209.10015 [cs.LG].

27. Swayamdipta, S. *et al.* Dataset Cartography: Mapping and Diagnosing Datasets with Training Dynamics. *CoRR* **abs/2009.10795.** arXiv: 2009.10795. https://arxiv.org/abs/2009.10795 (2020).

28. Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y. & Bresson, X. *Graph Neural Networks with Learnable Structural and Positional Representations* in *International Conference on Learning Representations* (2022). https://openreview.net/forum?id=wTTjnvGphYj.

29. Akujuobi, U., Han, Y., Zhang, Q. & Zhang, X. Collaborative Graph Walk for Semi-supervised Multi-Label Node Classification. *CoRR* **abs/1910.09706.** arXiv: 1910.09706. http://arxiv.org/abs/1910.09706 (2019).

30. Haykin, S. *Neural networks: a comprehensive foundation* (Prentice Hall PTR, 1994).

31. Perozzi, B., Al-Rfou, R. & Skiena, S. *Deepwalk: Online learning of social representations* in *Proc. of the International Conference on Knowledge Discovery and Data Mining* (2014).

32. Zhu, J. *et al.* Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. *Advances in Neural Information Processing Systems* **33** (2020).

33. Maurya, S. K., Liu, X. & Murata, T. *Improving Graph Neural Networks with Simple Architecture Design* 2021. arXiv: 2105.07634 [stat.ML]. https://arxiv.org/abs/2105.07634.

34. Dong, T. N. & Khosla, M. *Towards a consistent evaluation of miRNA-disease association prediction models* in *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2020), 1835–1842.

35. Yang, Y., Lichtenwalter, R. N. & Chawla, N. V. Evaluating link prediction methods. *Knowledge and Information Systems* **45,** 751–782 (2015).

4

AGALE: A Graph-Aware Continual Learning Evaluation Framework

[0]This chapter is based on the publication: **Tianqi Zhao**, Alan Hanjalic, Megha Khosla, *AGALE: A Graph-Aware Continual Learning Evaluation Framework*, *Transactions on Machine Learning Research (TMLR)*, Learning on Graphs Conference (LoG), 2024.

4.1. INTRODUCTION

Continual Learning (CL) describes the process by which a model accumulates knowledge from a sequence of tasks while facing the formidable challenge of preserving acquired knowledge amidst data loss from prior tasks. It finds application in several fields, such as the domain of medical image analysis, where a model has to detect timely emerging new diseases in images while maintaining the accuracy of diagnosing the diseases that have been encountered in the past. Significant achievements have been made on CL for euclidean data domains such as images and text [1–5]. Recent works have also delved into the broader scenario of multi-label continual learning (MLCL) [6–9], where one instance can be simultaneously associated with multiple labels.

To foster fair evaluation and identify new challenges in CL settings, several evaluation frameworks [10, 11] have been proposed, focusing on the single- and multi-label classification task on the euclidean data. However, these frameworks are not trivially applicable to graph-structured data due to the complexities arising from interconnections and multi-label nodes within graphs. Besides, existing evaluation frameworks in continual graph learning (CGL) [12, 13] evaluate the node classification task in the setting of associating nodes with a single label (which we refer to as the *single-label* scenario), thereby overlooking the possibility for nodes from previous tasks to adopt different labels in new tasks or acquire additional labels with time. For instance, in the context of a dynamically evolving social network, not only can new users with diverse interests (labels) be introduced over time, but existing users may also lose old labels or accumulate new labels continuously.

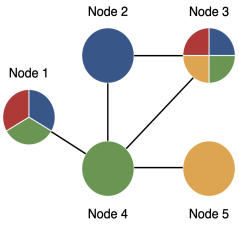


Figure 4.1: An example multi-label graph with colors indicating to the different node labels.

To illustrate the limitations of current CL evaluation frameworks when considering the multi-label scenario in graphs, we start with an example of a multi-label graph as in Figure 4.1. We use color coding to indicate the classes the nodes belong to. Please note that in what follows, we employ the term "class" to refer to the classes that correspond to a task. To refer to a class assigned to a particular node we use the term "label".

4.1.1. LIMITATIONS OF CURRENT CONTINUAL LEARNING EVALUATION FRAMEWORKS

Lack of graph-aware data partitioning strategies. Current experimental setups typically simulate continual learning settings by employing certain data partitioning methods over static data. However, existing CGL frameworks do not consider the multi-label scenario in the data partitioning algorithms.

The multi-label continual learning evaluation framework for Euclidean data [14] suggest the use of *hierarchical clustering* techniques, which involves grouping classes into a single task based on their co-occurrence frequency and subsequently eliminating instances with label sets that do not align with any class groups. Applying such a technique to graph-structured data not only risks excluding nodes with a higher number of labels but also disrupts the associated topological structures.

In Figure 4.2, we illustrate an application of the existing MLCL framework to the multi-label graph depicted in Figure 4.1. The classes blue, green, and red are collectively treated as one task due to their frequent co-occurrence. Node 3, having the maximum number of labels, is removed from the graph since no task encompasses all its labels. It is noteworthy that employing hierarchical clustering techniques increases the likelihood of eliminating nodes with more labels, effectively reducing both the number of multi-labeled nodes and the associated topological structure. In the current example, proper training and testing of the model on the red class is hindered, as only one node remains in the subgraph with the label red. Besides, node 5 becomes isolated in the second subgraph.

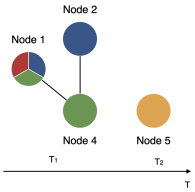


Figure 4.2: Subgraphs generated by grouping frequently co-occurring classes as a task.

Generation of train/val/test sets. Furthermore, the data partitioning algorithm is also responsible for the division of each subgraph into training, validation, and test subsets. In Figure 4.3 we show an example of train/val/test subsets generated using the strategy adopted by previous CGL evaluation frameworks for the task of distinguishing between blue and green classes. In particular, nodes belonging to a single class are divided independently at random into train/validation/test sets, assuming no overlap between classes. However, when each node can belong to multiple classes, an independent and random division within each class is not always feasible. For instance, the same node may serve as training data for one class and testing data for another in the same task, as is the case for node 1 in Figure 4.3. In this particular case, the model may observe the test data during the training process, resulting in data leakage. Conversely, considering the entire dataset as a whole for division would result in the dominance of the larger class, causing all nodes from the smaller class to be aggregated into the same split and thereby under-representing the smaller class in the other split. For instance, in multi-label datasets such as YELP, two classes can exhibit complete overlap, where all nodes from the smaller class also belong to the larger class. In this scenario, if the nodes belonging to the larger class are split first, there might be no nodes left to make the required splits for the smaller class.

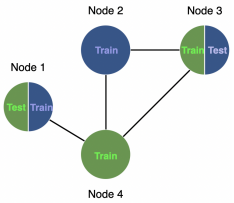


Figure 4.3: The split of the nodes within one subgraph generated by the previous CGL framework.

Use of predefined class orders. Existing CGL evaluation frameworks rely on a single predefined class order in the dataset and group sets of K classes as individual tasks. As the predefined order might not always reflect the true generation process of the data, it is important to evaluate CL models over several random class orders. Specifically for multi-label scenarios, the employed class order may not only influence the nodes and their neighborhood structures presented at each time step but also affect the number of labels assigned to a particular node in a given task. We demonstrate in Figure 4.4, using nodes from the multi-label graph in Figure 4.1, how distinct class orders generate subgraphs with the same set of nodes but with different topologies and node label assignments.

Limitations on the number of tasks. Last but not least, previous CGL benchmarks typically predetermined the size of model's output units, assuming an approximate count of classes in each graph during model initialization. However, this assumption is unrealistic because the eventual class size is often unknown, leading to potential inefficiencies in storage or capacity overflow.

4.1.2. OUR CONTRIBUTIONS

To tackle the above-mentioned gaps, we develop a generalized evaluation framework for continual graph learning, which is applicable both for multi-class and multi-label node classification tasks and can be easily adapted for multi-label graph and edge classification tasks. Our key contributions are as follows.

- We define two generalized incremental settings for the node classification task in the CGL evaluation framework which are applicable for both multi-class and multi-label datasets.
- We develop new data split algorithms for curating CGL datasets utilizing existing static benchmark graph datasets. We theoretically analyze the *label homophily* of the resulting subgraphs which is an important factor influencing the performance of learning models over graphs.
- We perform extensive experiments to assess and compare the performance of well-established methods within the categories of Continual Learning (CL), Dynamic Graph Learning (DGL), and Continual Graph Learning (CGL). Through our analysis, we evaluate these methods in the context of their intended objectives, identifying their constraints and highlighting potential avenues for designing more effective models to tackle standard tasks in CGL.
- We re-implement the compared models in our framework while adapting them for the unknown number of new tasks that may emerge in the future.

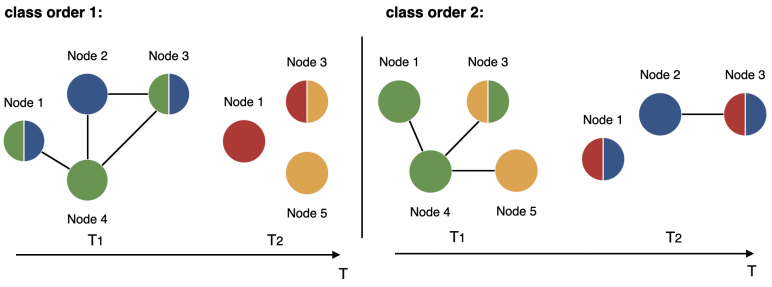


Figure 4.4: An example of subgraphs obtained by applying different class orders for the static multi-label graph in Figure 4.1.

4.2. PROBLEM FORMULATION

We start by providing a general formulation of the continual learning problem for graph-structured data and elaborate on the additional complexities when the nodes in the graph may have multiple labels as compared to the single-label scenario.

Problem Setting and Notations. *Given a time sequence $\mathcal{T} = \{1, 2, \ldots, T\}$, at each time step $t \in \mathcal{T}$, the input is one graph snapshot $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{X}_t, \mathbf{Y}_t)$, with node set $\mathcal{V}_t$ and edge set $\mathcal{E}_t$. Additionally, $\mathbf{X}_t \in \mathcal{R}^{|\mathcal{V}_t| \times D}$ and $\mathbf{Y}_t \in \{0,1\}^{|\mathcal{V}_t| \times |\mathcal{C}_t|}$ denote the feature matrix and the label matrix for the nodes in graph $\mathcal{G}_t$, where D is the dimension of the feature vector, and $\mathcal{C}_t$ is the set of classes seen/available at time t. We assume that the node set $\mathcal{V}_t$ is partially labeled, i.e., $\mathcal{V}_t = \{\mathcal{V}_t^l, \mathcal{V}_t^u\}$, where $\mathcal{V}_t^l$ and $\mathcal{V}_t^u$ represent the labeled nodes and the unlabeled nodes in $\mathcal{G}_t$. We use $\mathbf{A}_t$ to denote the adjacency matrix of $\mathcal{G}_t$. We use $\mathcal{Y}^v$ to denote the complete label set of node v and $\mathcal{Y}_t^v$ to denote the label set of node v observed at time t.*

Objective. *The key objective in CGL setting, as described above, is to predict the corresponding label matrix of $\mathcal{V}_t^u$ denoted by $\mathbf{Y}_t^u$ (when the complete label set is restricted to $\mathcal{C}_t$) while maintaining the performance over the classification on nodes in all graphs in the past time steps in $\{1, 2, \ldots, t-1\}$.*

4.2.1. DIFFERENCES TO SINGLE-LABEL SCENARIO

Having explained the problem setting and the objective we now describe the key differences of the multi-label scenario as compared to the single-label case in continual graph learning, which were so far ignored by previous works resulting in various limitations as illustrated in Section 4.1.1.

- **Node overlaps in different tasks.** In the single-label scenario each node is affiliated with one single class, exclusively contributing to one task. The following statement, which states that no node appears in more than one task, always holds:

$$\forall i, j \in \mathcal{T}, \text{and } i \neq j, \mathcal{V}_i \cap \mathcal{V}_j = \emptyset \tag{4.1}$$

  However, in the multi-label scenario, one node can have multiple labels and can therefore participate in multiple tasks as time evolves. Contrary to the single-label scenario, when the nodes have multiple labels, there will exist at least a pair of tasks with at least one node in common as stated below.

$$\exists i, j \in \mathcal{T}, \text{and } i \neq j, \mathcal{V}_i \cap \mathcal{V}_j \neq \emptyset \tag{4.2}$$

- **Growing label sets.** In the single-label scenario, the label set of a node v, $\mathcal{Y}^v$, stays the same across different time steps, i.e.,

$$\forall i, j \in \mathcal{T}, \mathcal{Y}_i^v = \mathcal{Y}_j^v \tag{4.3}$$

  However, in the multi-label scenario, the label set of a node may grow over time, i.e., a node may not only appear in several tasks as above but also have different associated label sets, i.e., the following holds.

$$\exists i, j \in \mathcal{T}, \mathcal{Y}_i^v \neq \mathcal{Y}_j^v \tag{4.4}$$

- **Changing node neighborhoods.** Note that while simulating a continual learning scenario, subgraphs are curated corresponding to sets of classes/labels required to be distinguished in a particular task. In other words, the subgraph presented for a particular task only contains edges connecting nodes with the label set seen in that task. Therefore, the neighborhood of a node v, denoted as $\mathcal{N}^v$ can also be different across different time steps in the multi-label scenario, i.e.,

$$\exists i, j \in \mathcal{T}, \mathcal{N}_i^v \neq \mathcal{N}_j^v \tag{4.5}$$

In the multi-label graphs, both multi-label and single-label nodes exist, providing therefore a suitable context to develop a generalized CGL evaluation framework as elaborated in the next section.

4.3. AGALE: OUR EVALUATION FRAMEWORK

We present a holistic continual learning evaluation framework for graph-structured input data, which we refer to as AGALE (a graph-aware continual learning evaluation). We begin by developing two generalized incremental settings (in Section 4.3.1) that accommodate the requirements of the multi-label scenario (as discussed in Section 4.2.1) with respect to node overlaps in different tasks and growing label sets. In Section 4.3.2, we develop new data partitioning algorithms designed to derive subgraphs and training partitions from a static graph dataset, tailored specifically for the developed incremental settings. To underscore the significance of our approach, we provide theoretical analysis of AGALE in Section 4.3.3 and compare it with the previously established CGL and MLCL frameworks in Section 4.3.4.

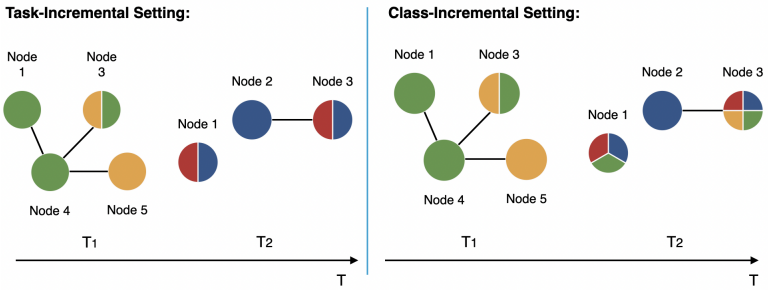


Figure 4.5: Visualization of our proposed generalized evaluation CGL framework AGALE.

4.3.1. TWO GENERALIZED INCREMENTAL SETTINGS FOR CONTINUAL GRAPH LEARNING

We *first* define and develop two generalized incremental settings in CGL, i.e., TASK-IL SETTING and CLASS-IL SETTING. **In the TASK-IL SETTING**, the goal is to distinguish between classes specific to each task. Different from single-label settings, the multi-labeled nodes can be present with non-overlapping subsets of their labels in different subgraphs, as shown in Figure 4.5. Formally, for any node v in the multi-label graph, in the TASK-IL

SETTING we have

$$\forall i, j \in \mathcal{T}, \mathcal{Y}_i^v \cap \mathcal{Y}_j^v = \emptyset.$$

In the CLASS-IL SETTING, the goal is to distinguish among all the classes that have been seen so far. Specifically, in addition to the same node appearing in multiple tasks as in the previous setting, a node with multiple labels can attain new labels as new tasks arrive, as shown in Figure 4.5. Formally, for any node v in the multi-label graph,

$$\forall i, j \in \mathcal{T}, \text{if } i < j, \text{ then } \mathcal{Y}_i^v \subseteq \mathcal{Y}_j^v$$

Note that the above settings allow for node overlaps and growing/changing label sets of the same node at different time points.

4.3.2. DATA PARTITIONING ALGORITHMS

We now describe our data partitioning algorithms to simulate sequential data from static graphs. The design strategy of our algorithms takes into account of the node overlaps in tasks, the growing/changing label set of nodes over time, and the changing node neighborhoods while minimizing the loss of node labels and the graph's topological structure during partitioning. Our developed data partition strategy can be employed in both incremental settings and consists of the following two main parts.

- **Task sequence and subgraph sequence generation.** We employ Algorithm 2 to segment the provided graph from the dataset into coherent subgraph sequences. We first remove the classes with size smaller than a threshold δ. Instead of using a predefined class order (as discussed in Section 4.1.1) we generate n random orders of the remaining classes to simulate the random emergence of new classes in the real world. Specifically, given a dataset with C classes, we group K random classes as one task for one time step. At any time point t, let $\mathcal{C}_t$ denote the set of classes grouped for the task at time t. We construct a subgraph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ such that $\mathcal{V}_t$ is the set of nodes with one or more labels in $\mathcal{C}_t$. The edge set $\mathcal{E}_t$ consists of the induced edges on $\mathcal{V}_t$. Note that the number of classes chosen to create one task is adaptable. In order to maximize the length of the task sequence for each given graph dataset and subsequently catastrophic forgetting, we choose $K = 2$ in this work. If the dataset has an uneven number of classes in total, the remaining last class will be grouped with the second last class group.
- **Construction of train/val/test sets.** To overcome the current limitations of generating train/val/test sets as discussed in Section 4.1.1, we employ Algorithm 3 to partition nodes of a given graph snapshot $\mathcal{G}_t$. For the given subgraph $\mathcal{G}_t$, our objective is to maintain the pre-established ratios for training, validation, and test data for both the task as a whole and individual classes within the task. To achieve this, our procedure starts with the determination of the size of each class. Note that the cumulative sizes of these classes may exceed the overall number of nodes in the subgraph due to multi-labeled nodes being accounted for multiple times based on their respective labels. Subsequently, the classes are arranged in ascending order of size, starting with the smallest class. The smallest class is partitioned in accordance with the predefined proportions. Subsequent classes in the order undergo partitioning with the following steps:
  - We identify nodes that have already been allocated to prior classes.

- We then compute the remaining node counts for the training, validation, and test sets in accordance with the predefined proportions for the current class.
- Finally, we split randomly the remaining nodes within the current class into train/-val/test sets such that their predefined proportions are respected.

Note that for a given class order, the structural composition of each subgraph remains constant across both the incremental settings. What distinguishes these incremental settings is the label vector assigned to the nodes. Specifically, nodes with a single label manifest uniquely in one subgraph corresponding to a task. Conversely, nodes with multiple labels appear in the TASK-IL SETTING with distinct non-overlapping subsets of their original label set across various subgraphs while appearing with the expansion of their label vectors in the CLASS-IL SETTING.

Algorithm 2 Task Sequence and Subgraph Sequence Generation

Require: Static graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with classes $\mathcal{C} = \{c_1, c_2, \ldots, c_C\}$, threshold of small classes δ, group size K

Ensure: n task sequences $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n\}$ and for each task sequence $\mathcal{S}_i$ a corresponding subgraph sequence $\mathcal{G}_i = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_T\}$

for $c_j \in \mathcal{C}$ **do**
  $\mathcal{V}_{c_j} = \{v_i | c_j \in y_i\}$
$\mathcal{C}' = \{\mathcal{C} - c_j || \mathcal{V}_{c_j}| < \delta\}$
Generate n random orders of $\mathcal{C}'$: $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_n\}$
for $\mathcal{O}_j \in \mathcal{O}$ **do**
  for $t = 1$ to $\lfloor \frac{C}{k} \rfloor = T$ **do**
    Group the first k classes as a task: $\mathcal{S}_t = \{c_1, \ldots, c_k\}$
    $\mathcal{O}_j = \mathcal{O}_j - \mathcal{S}_t$
    $\mathcal{V}_t = \{v_i | \mathbf{y_i} \cap \mathcal{S}_t \neq \emptyset\}$
    $\mathcal{E}_t = \{e(u, v) | e \in \mathcal{E} \wedge u, v \in \mathcal{V}_t\}\}$
    $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$

In the Appendix 4.8.1, we present an analysis of the subgraphs derived by AGALE from the given static graph in PCG as an example of showcasing the efficacy of our approach.

4.3.3. THEORETICAL ANALYSIS OF AGALE

As studied in previous works [15, 16], the similarity of labels between neighboring nodes (usually termed label homophily) influences the performance of various graph machine learning algorithms for the task of node classification in the static case. We here provide a theoretical analysis of AGALE with respect to the label homophily of generated subgraphs under different conditions. We would later use our theoretical insights and the dataset properties to analyze the performance of various methods. We use the following definition of label homophily for multi-label graphs proposed in [16].

Algorithm 3 Train and Test Partition Algorithm Within One Subgraph

Require: subgraph $\mathcal{G}_t$ in subgraph sequence $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_T\}$, proportion set P for train, validation, and test $P = \{P_{train}, P_{val}, P_{test}\}$
Ensure: the split within subgraph $\mathcal{G}_t = \{\mathcal{V}_t^{train}, \mathcal{V}_t^{val}, \mathcal{V}_t^{test}\}$ for task $\mathcal{S}_t$

Get the classes for the current task $\mathcal{S}_t = \{c_1, \ldots, c_k\}$
$\mathcal{O}' = Sort_{ascend}(|\mathcal{V}_{c_j}|)$ for $c_j \in \mathcal{S}_t$
initialize empty node set $\mathcal{V}_t^{train}$, $\mathcal{V}_t^{val}$, and $\mathcal{V}_t^{test}$
initialize empty encountered nodes set $\mathcal{V}_t$
for $c \in \mathcal{O}'$ **do**
    $\mathcal{V}_c = \{v_i | c \in y_i\}$
    if c is the smallest class in $\mathcal{S}_i$ **then**
        Randomly split $\mathcal{V}_c$ into $\mathcal{V}_c^{train}$, $\mathcal{V}_c^{val}$, $\mathcal{V}_c^{test}$ according to P
    else
        Calculate the size of train/val/test set $|\mathcal{V}_c^{train}|$, $|\mathcal{V}_c^{val}|$, $|\mathcal{V}_c^{test}|$ according to P
        $\mathcal{V}_t^{dup} = \mathcal{V}_c \cap \mathcal{V}_t$
        $\mathcal{V}_c = \mathcal{V}_c - \mathcal{V}_t^{dup}$
        for $v_i \in \mathcal{V}_{dup}$ **do**
            for $split \in [\mathcal{V}_c^{train}, \mathcal{V}_c^{val}, \mathcal{V}_c^{test}]$ **do**
                if v_i in $split$ **then**
                    $|split| = |split| - 1$
        for $split \in [\mathcal{V}_c^{train}, \mathcal{V}_c^{val}, \mathcal{V}_c^{test}]$ **do**
            Randomly choose $|split|$ nodes from $\mathcal{V}_c$ to add to $split$
    add $\mathcal{V}_c^{train}$, $\mathcal{V}_c^{val}$, $\mathcal{V}_c^{test}$ to $\mathcal{V}_t^{train}$, $\mathcal{V}_t^{val}$, $\mathcal{V}_t^{test}$
    add $\mathcal{V}_c$ to $\mathcal{V}_t$

Definition 5. *Given a multi-label graph $\mathcal{G}$, the label homophily h of $\mathcal{G}$ is defined as the average of the Jaccard similarity of the label set of all connected nodes in the graph:*

$$h = \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} \frac{|\mathcal{Y}^i \cap \mathcal{Y}^j|}{|\mathcal{Y}^i \cup \mathcal{Y}^j|}$$

Let for any two connected nodes $i, j \in \mathcal{V}$, $h_{\mathcal{G}}^{e(i,j)}$ denotes the label homophily over the edge $e(i,j) \in \mathcal{E}$ in graph $\mathcal{G}$. We then have the following result about the label homophily of $e(i,j)$ in the subgraph $\mathcal{G}_t$ generated by AGALE at time t.

Theorem 2. *For any edge $e(i,j) \in \mathcal{E}$ and any subgraph at time t, $\mathcal{G}_t$ such that $e(i,j) \in \mathcal{E}_t$, $h_{\mathcal{G}_t}^{e(i,j)} \geq h_{\mathcal{G}}^{e(i,j)}$ when at least one of the nodes in $\{i,j\}$ is single-labeled. For the case when both nodes i, j are multi-labeled, we obtain $h_{\mathcal{G}_t}^{e(i,j)} \geq h_{\mathcal{G}}^{e(i,j)}$ with probability at least $(1 - (1 - h_{\mathcal{G}}^{e(i,j)})^K)$ for* TASK-IL SETTING *and $(1 - (1 - h_{\mathcal{G}}^{e(i,j)})^{Kt})$ for* CLASS-IL SETTING.

Proof. In the multi-label graphs, one pair of connected nodes belongs to the following three scenarios: 1) two single-labeled nodes are connected, 2) a single-label node is connected to a multi-labeled node, and 3) two multi-labeled nodes are connected.

Scenario 1: Note that at any time step t two nodes i and j are connected if and only if at least one label for each node appears in $\mathscr{C}_t$. As in the first scenario, both the nodes are single-labeled, and the label homophily score for edge $e(i,j)$ stays the same in the subgraph as in the original graph:

$$h_{\mathscr{G}_t}^{e(i,j)} = h_{\mathscr{G}}^{e(i,j)} = \begin{cases} 0, & \text{if } \mathscr{Y}^i \neq \mathscr{Y}^j \\ 1, & \text{if } \mathscr{Y}^i = \mathscr{Y}^j \end{cases} \tag{4.6}$$

Scenario 2: In the second scenario, where one single-labeled node i is connected to a multi-labeled node j, at any time step t, when $e(i,j)$ appears in the subgraph $\mathscr{G}_t$,

$$h_{\mathscr{G}_t}^{e(i,j)} \geq h_{\mathscr{G}}^{e(i,j)} \begin{cases} h_{\mathscr{G}_t}^{e(i,j)} = h_{\mathscr{G}}^{e(i,j)} = 0, & \text{if } \mathscr{Y}^i \notin \mathscr{Y}^j \\ h_{\mathscr{G}_t}^{e(i,j)} = \begin{cases} \frac{1}{2}, & \text{if } \mathscr{Y}^i \subset \mathscr{C}_t \cap \mathscr{Y}^j \\ 1, & \text{if } \mathscr{C}_t \cap \mathscr{Y}^j = \mathscr{Y}^i \end{cases} \geq h_{\mathscr{G}}^{e(i,j)}, & \text{if } \mathscr{Y}^i \in \mathscr{Y}^j \end{cases} \tag{4.7}$$

Combining (4.6) and (4.7) we note that when at least one node in an edge is single-labeled, the label homophily of the corresponding edge will be equal to more than that in the static graph, thereby completing the first part of the proof.

Scenario 3: In the third scenario, where two multi-labeled nodes i and j are connected, at any time step t, when $e(i,j)$ appears in the subgraph $\mathscr{G}_t$, it holds $\mathscr{C}_t \cap \mathscr{Y}_i \neq \emptyset$ and $\mathscr{C}_t \cap \mathscr{Y}_j \neq \emptyset$. In this scenario, the label homophily of an edge depends on the relationship between $\mathscr{Y}_i \cap \mathscr{Y}_j$ and $\mathscr{C}_t$:

$$\begin{cases} 0 = h_{\mathscr{G}_t}^{e(i,j)} < h_{\mathscr{G}}^{e(i,j)} & \text{if } \mathscr{Y}^i \cap \mathscr{Y}^j \cap \mathscr{C}_t = \emptyset \\ \begin{cases} h_{\mathscr{G}_t}^{e(i,j)} = \frac{1}{2} & \text{TASK-IL SETTING} \\ h_{\mathscr{G}_t}^{e(i,j)} \geq \frac{1}{2t} & \text{CLASS-IL SETTING} \end{cases} & \text{if } \mathscr{Y}^i \cap \mathscr{Y}^j \neq \emptyset,\ \mathscr{Y}^i \cap \mathscr{Y}^j \cap \mathscr{C}_t \subset \mathscr{Y}^i \cap \mathscr{Y}^j, \\ & \quad \mathscr{Y}^i \cap \mathscr{Y}^j \cap \mathscr{C}_t \subset \mathscr{C}_t \\ h_{\mathscr{G}_t}^{e(i,j)} \geq h_{\mathscr{G}}^{e(i,j)} & \text{if } \mathscr{Y}^i \cap \mathscr{Y}^j \subset \mathscr{C}_t \\ h_{\mathscr{G}_t}^{e(i,j)} = 1 \geq h_{\mathscr{G}}^{e(i,j)} & \text{if } \mathscr{C}_t \subseteq \mathscr{Y}^i \cap \mathscr{Y}^j \end{cases} \tag{4.8}$$

Note that all the statements hold in both incremental settings except for the second condition, where $\mathscr{Y}_t^i \cap \mathscr{Y}_t^j \cap \mathscr{C}_t$ is the strict subset of $\mathscr{Y}_t^i \cap \mathscr{Y}_t^j$ and $\mathscr{C}_t$. With a relatively smaller size of $|\mathscr{C}_t| = K = 2$ in our setting, we have in the TASK-IL SETTING, $|\mathscr{Y}_t^i \cap \mathscr{Y}_t^j| = 1$ and $|\mathscr{Y}_t^i \cup \mathscr{Y}_t^j| = 2$:

$$h_{\mathscr{G}_t}^{e(i,j)} = \frac{|\mathscr{Y}_t^i \cap \mathscr{Y}_t^j|}{|\mathscr{Y}_t^i \cup \mathscr{Y}_t^j|} = \frac{1}{2} \tag{4.9}$$

while in the CLASS-IL SETTING, because $|\mathscr{Y}_t^i \cap \mathscr{Y}_t^j| \geq 1$, $|\mathscr{Y}_t^i \cup \mathscr{Y}_t^j| \leq Kt$, we obtain

$$h_{\mathscr{G}_t}^{e(i,j)} = \frac{|\mathscr{Y}_t^i \cap \mathscr{Y}_t^j|}{|\mathscr{Y}_t^i \cup \mathscr{Y}_t^j|} \geq \frac{1}{2t} \tag{4.10}$$

We can now upper bound the probability of the worst case event, i.e., when an edge $e(i,j)$ exists at time t but $\mathscr{C}_t \cap \mathscr{Y}^i \cap \mathscr{Y}^j = \emptyset$. This can only happen if the classes in set

$\mathscr{C}_t$ are chosen from the set $\mathscr{Y}^i \cup \mathscr{Y}^j \setminus \mathscr{Y}^i \cap \mathscr{Y}^j$. For TASK-IL SETTING, the probability of choosing at least one element of $\mathscr{C}_t$ from the common labels of node i and j is equal to $h_{\mathscr{G}}^{e(i,j)}$. Then the probability that none of the classes in $\mathscr{C}_t$ appear in the common set is at most $(1 - h_{\mathscr{G}}^{e(i,j)})^{|\mathscr{C}_t|}$. The proof is completed by noting the fact that $|\mathscr{C}_t| = K$ for TASK-IL SETTING and $|\mathscr{C}_t| = Kt$ for CLASS-IL SETTING at time step t. □

4.3.4. COMPARISON WITH PREVIOUS EVALUATION FRAMEWORKS

In response to overlooked challenges in established CGL and MLCL evaluation frameworks, as detailed in Section 4.1, our framework tackles these issues by the following.

- **Incorporation of the multi-label scenario.** Contrary to previous evaluation frameworks AGALE accommodates single-label and multi-label node nodes in the following ways.
  - For single-label nodes, our framework expands upon previous methods during the task sequence's creation phase. It introduces dynamics in label correlations by allowing random class ordering to generate the task sequence. This results in diverse subgraph sequences, mimicking the random emergence of new trends in the real world.
  - Regarding the multi-label scenario, as shown in Figure 4.5, our framework allows for update/change of label assignments for a given node in the TASK-IL SETTING and expansion of the node's label set in the CLASS-IL SETTING.
- **Information preservation and prevention of data leakage**
  - As described in Section 4.3.2, the data partitioning strategies of AGALE ensure that no nodes from the original multi-label static graph are removed while creating the tasks. Single-labeled nodes appear once in the task sequence in both settings, while multi-labeled nodes surface with different labels in TASK-IL SETTING and CLASS-IL SETTING. Specifically, they appear with non-overlapping subsets of their label set in TASK-IL SETTING, and as the class set expands, their entire label set is guaranteed to be seen by the model before the final time step in CLASS-IL SETTING.
  - Previous CGL evaluation frameworks split the nodes into train and evaluation sets within each class, not considering the situation where one node can belong to multiple classes in the task. Such a strategy may lead to data leakage as one node can be assigned to training and testing sets for the same task. During task training on a subgraph comprising various classes, our framework ensures no overlap among the training, validation, and test sets. Single-labeled nodes exclusively belong to one class, preventing their re-splitting after the initial allocation. For multi-label nodes that have been allocated to a particular class (see lines 11 and 12 in Algorithm 3), we exclude them from the remaining nodes of other classes they belong to, eliminating any potential data leakage during training and evaluation within one subgraph.
  - In addition, we approach the continual learning setting by not allowing the inter-task edges. This deliberate choice means that, upon the arrival of a new task, the model no longer retains access to the data from the previous time steps.

- **Ensuring fair split across different classes and the whole graph.** Due to the differences in the class size, a split from the whole graph will result in the bigger class dominating the splits, leaving the small class underrepresented in the splits. Moreover, the split within each class may result in data leakage in one subgraph, as explained in the previous paragraph. To maintain a fair split despite differences in class sizes, our framework prioritizes splitting smaller classes initially. It subsequently removes already split nodes from larger class node sets. This approach guarantees an equitable split within each class and from within the whole subgraph, preventing larger classes from dominating the splits and ensuring adequate representation for smaller classes.

- **Application for graph/edge-level CGL.** AGALE can be directly applied for the graph classification task, each input data is an independent graph without interconnections. For the edge classification task, our framework can be applied by first transforming the original graph $\mathcal{G}$ into a line graph $L(\mathcal{G})$, where for each edge in $\mathcal{G}$, we create a node in $L(\mathcal{G})$; for every two edges in $\mathcal{G}$ that have a node in common, we make an edge between their corresponding nodes in $L(\mathcal{G})$.

4.4. RELATED WORK

4.4.1. CONTINUAL LEARNING

Continual Learning [4, 17–22], a fundamental concept in machine learning, addresses the challenge of enabling models to learn from and adapt to evolving data streams over time. Continual learning has applications in a wide range of domains, including computer vision, natural language processing, and reinforcement learning, making it an active area of research with practical implications for the lifelong adaptation of machine learning models. Unlike traditional batch learning, where models are trained on static datasets, continual learning systems aim to learn from new data while preserving previously acquired knowledge sequentially. This paradigm is particularly relevant in real-world scenarios where data is non-stationary and models need to adapt to changing environments.

The key objectives of continual learning are to avoid catastrophic forgetting, where models lose competence in previously learned tasks as they learn new ones, and to ensure that the model's performance on earlier tasks remains competitive. Various techniques have been proposed in the literature to tackle these challenges, which can be categorized into four categories.

- **Knowledge distillation methods.** The methods from this category [6, 7, 20] retain the knowledge from the past by letting the new model mimic the old model on the previous task while adapting to the new task. Overall, the learning objective can be summarized as to minimize the following loss function:

$$\mathscr{L} = \lambda_o \mathscr{L}_{\text{old}}\left(\mathbf{Y}_o, \hat{\mathbf{Y}}_o\right) + \mathscr{L}_{\text{new}}\left(\mathbf{Y}_n, \hat{\mathbf{Y}}_n\right) + \mathscr{R}, \tag{4.11}$$

  where $\mathscr{L}_{\text{old}}$ and $\mathscr{L}_{\text{new}}$ represent the loss functions corresponding to the old and new tasks, respectively. The parameter λ_o is the weight for balancing the losses, and $\mathscr{R}$ encapsulates the regularization term. The process of transferring knowledge from a pre-existing model (teacher) to a continually evolving model (student) in knowledge distillation unfolds within $\mathscr{L}_{\text{old}}$, where the new model undergoes training to align its predictions on new data for

the old task, denoted as $\hat{\mathbf{Y}}_o$, with the predictions of the previous model on the same new data for the old task, represented as $\mathbf{Y}_o$. Simultaneously, the new model approximates its prediction of the new data on the new task $\hat{\mathbf{Y}}_n$ to their true labels $\mathbf{Y}_n$. For example, LwF [20] minimize the difference between the outputs of the previous model and the new model on the new coming data for the previous tasks while minimizing the classification loss of the new model on the new task.

- **Regularization strategies.** The methods in this category maintain the knowledge extracted from the previous task by penalizing the changes in the parameters θ of the model trained for the old tasks. Typically, the following loss is minimized:

$$\mathscr{L}(\theta) = \mathscr{L}_{\text{new}}(\theta) + \lambda \sum_i \Omega_i \left(\theta_{\mathbf{i}} - \theta_{\mathbf{i}}{}^*\right)^2 \tag{4.12}$$

  where $\mathscr{L}_{\text{new}}$ denotes the loss function for the new task, θ is the set of model parameters. The parameter λ functions as the weight governing the balance between the old and new tasks, while Ω_i represents the importance score assigned to the ith parameter $\theta_{\mathbf{i}}$. For example, MAS [21] assigns importance scores for the parameters by measuring how sensitive the output is to the change of the parameters. The term θ_i^* refers to the prior task's parameter determined through optimization for the previous task.

- **Replay mechanisms.** Methods from this category extract representative data from the previous tasks and employ them along with the new coming data for training to overcome catastrophic forgetting [23, 24]. Methods under this category mainly differ with respect to their approaches to sampling representative data from the old task for storage in the buffer. For example, Kim *et al.* [24] maintains a target proportion of different classes in the memory to tackle the class imbalance in the multi-label data.

- **Dynamic architectures.** Methods from this category [9, 25] dynamically expand their architecture when needed for new tasks. This expansion may include adding new layers or neurons to accommodate new knowledge. For example, Lee *et al.* [25] dynamically expands the network architecture based on the relevance between new and old tasks.

Another line of work in CL focuses on benchmarking evaluation methods. For instance, Farquhar & Gal [10] and Lange *et al.* [11] provide more robust and realistic evaluation metrics for the CL methods, incorporating real-world challenges like varying task complexities and the stability gap.

4.4.2. CONTINUAL GRAPH LEARNING

As a sub-field of continual learning, Continual Graph Learning (CGL) addresses the catastrophic forgetting problem as the model encounters new graph-structured data over time. Within CGL, two primary lines of work exist. The first involves establishing evaluation frameworks that define incremental settings in CGL scenarios and their corresponding data partitioning algorithms. The second line of work focuses on proposing new methods based on specific predefined CGL incremental settings derived from these evaluation frameworks. Our work mainly falls into the first category in which we develop a more holistic evaluation framework covering the multi-label scenario for graph-structured data.

The previously established CGL frameworks focus on benchmarking tasks in CGL. For instance, [13] defined Task- and Class- Incremental settings for single-labeled node and graph classification tasks in CGL and studied the impact of including the inter-task edges among the subgraph. [12] expanded this by adding the domain- and time-incremental settings and including the link prediction task in the CGL benchmark. Additionally, surveys like [26] and [27] focus on categorizing the approaches in CL and CGL.

However, none of the above works sufficiently addressed the complexities of defining the multi-label node classification task within the CGL scenario. The only exception is Ko *et al.* [12], which used a graph with multi-labeled nodes, but that too in a domain incremental setting. In particular, each task was constituted of nodes appearing from a single domain. Consequently, a node appears in one and only one task together with all of its labels. This does not cover the general multi-label scenario in which the same node can appear in multiple tasks each time with different or expanding label sets.

Existing methods for CGL focus mainly on the multi-class scenario and fall into one of the four categories (see the previous subsection) of continual learning methods. For example, GRAPHSAIL [28] is a knowledge distillation approach that distills each node's local and global structure and its self-embedding knowledge, respectively. Regularization approach TWP [29] adds a penalization to the parameters that are important to the learned topological information in addition to the task-related loss to stabilize the parameters playing pivotal roles in the topological aggregation. ERGNN [30] is based on the replay mechanism and carefully selects nodes from the old tasks to the buffer and replays them with the new graph. [31] combines replay and regularization to preserve existing patterns.

4.4.3. LEARNING ON DYNAMIC GRAPHS

Since streaming graphs find applications in various domains, including social network analysis, recommendation systems, fraud detection, and knowledge graph refinement, several methods [32–35] have been proposed in the field of dynamic graph learning (DGL) to utilize the knowledge from the past to enhance the model's performance on the graph in the current timestamp. For example, Rossi *et al.* [36] uses the memory unit to represent the node's history in the compressed format, and Pareja *et al.* [37] uses recurrent architecture between the models trained for the adjacent time steps to let the new model inherent knowledge extracted from the old tasks. However, the designing goal of the methods in DGL is to utilize the knowledge extracted from the old tasks to enhance the performance of the model on the current task, while in CGL, we focus on the catastrophic forgetting problem, i.e., the model needs not only to perform well on the current task but also on the previous tasks in the task sequence. We compare and analyze the models from these two categories in detail in Section 4.6.

4.4.4. APPLICATION OF GRAPH MACHINE LEARNING IN CONTINUAL LEARNING

Some work [3, 38] also attempts to use graph structures to alleviate catastrophic forgetting in Euclidean data. For instance, Tang & Matteson [3] augments independent image data in memory with a learnable random graph, capturing similarities among them to alleviate catastrophic forgetting. However, as our current focus is solely on graph-structured data, these endeavors fall beyond the scope of this study.

4.5. EXPERIMENT SETUP

In this section, we test the state-of-art models from CL, DGL, and CGL domains. Note that in this study, we employ $P = 3$, indicating that we generate three random orders for the classes in each dataset in the experimental section. We introduce the models according to their categories.

4.5.1. METHODS

This subsection introduces all the methods used in the experiment section. The CL methods use Graph Convolutional Network (GCN) [39] as the backbone.

- **SIMPLEGCN**: We train GCN on each of the subgraph sequences without any continual learning technique, which is denoted as SIMPLEGCN in the following sections.
- **JOINTTRAINGCN**: We also include GCN trained on all the tasks simultaneously and therefore should not have the catastrophic forgetting problem. This setting is referred to as JOINTTRAINGCN in the following section.
- **Continual Learning Methods**: We choose Learning Without Forgetting (LWF), Elastic Weight Consolidation (EWC), and Memory Aware Synapses (MAS) from this category. LWF distill the knowledge from the old model to the new model to prevent the model from catastrophic forgetting. EWC and MAS are both regularization-based methods. The difference is that EWC penalizes the changes in the parameters that are important to the previous task, while MAS measures the importance of the parameters based on the sensitivity of the output on the parameters.
- **Dynamic Graph Neural Network**: We choose EVOLVEGCN [37] from this category, which uses recurrent architecture between the models trained for the adjacent time steps to let the new model inherent knowledge extracted from the old tasks to enhance the model's performance on the current task.
- **Continual Graph Learning Methods**: We choose ERGNN [30] from this category, which samples representative nodes from the old tasks in the buffer and replays them with the new data to address the catastrophic forgetting problem.

4.5.2. DATASETS

We demonstrate our evaluation framework on 3 multi-label datasets in this work. We also include 1 multi-class dataset CORAFULL as an example to demonstrate the generalization of our evaluation framework on single-label nodes. We include the description of the CORAFULL and the results on it in the Appendix 4.8.2.

The inter-task edges are defined in [13] as the edges that connect the new subgraph to the overall graph. We do not allow inter-task edges in our evaluation framework, i.e., at each time step, only the subgraph for the new task is used as input. The reason is that in CL, the assumption is that the model loses access to the data from the previous time steps. With the inter-task edges, the node features from the previous time step would also be used as input, which violates this assumption and alleviates the forgetting problem.

Below, we introduce the datasets used in this work:

1. PCG[16], in which nodes are proteins and edges correspond to the protein functional interaction, and the labels the phenotype of the proteins.

2. DBLP[40], in which nodes represent authors and edges the co-authorship between the authors, and the labels indicate the research areas of the authors.

3. YELP[41], in which nodes correspond to the customer reviews and edges to their friendships with node labels representing the types of businesses.

The statistics about the datasets are summarized in Table 4.1. We use the label homophily defined for multi-label graphs in Zhao *et al.* [16]. Following the application of a data partitioning algorithm, the given static graphs by the datasets are split into subgraph sequences. We also summarize the characteristics of the subgraphs to provide insights into the partitioned structure.

Table 4.1: The data statistics. Specifically, $|\mathcal{V}|$, $|\mathcal{E}|$, $|\mathcal{C}|$, $\overline{|\mathcal{L}|}$, and r_{homo} denote the number of nodes, edges, classes, mean label count per node, and label homophily of the static graph given by the dataset, respectively. $|T|$ signifies the count of tasks in the resulting task sequence. Additionally, $\overline{|\mathcal{V}|}$ and $\overline{|\mathcal{E}|}$ represent the average number of nodes and edges in a subgraph. Further details on label homophily are captured through $\overline{|r|}_{tsk}$ and $\overline{|r|}_{cls}$, representing the averaged label homophily of subgraphs in the TASK-IL SETTING and CLASS-IL SETTING), respectively.

	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{C}\|$	$\overline{\|\mathcal{L}\|}$	$\|T\|$	r_{homo}	$\overline{\|\mathcal{V}\|}$	$\overline{\|\mathcal{E}\|}$	$\overline{\|r\|}_{tsk}$	$\overline{\|r\|}_{cls}$
PCG	$3K$	$37K$	15	1.93	7	0.17	808	4763	0.64	0.38
DBLP	$28K$	$68K$	4	1.18	2	0.76	$15K$	$37K$	0.86	0.81
YELP	$716K$	$7.34M$	100	9.44	50	0.22	$121K$	$921K$	0.75	0.47

In Theorem 2 we theoretically analyzed the label homophily of the edges in the subgraphs where we showed that in cases of single-labeled nodes and for higher homophily edges, the homophily in subgraphs typically increases. Table 4.1 further shows that the average label homophily of the subgraphs is in fact higher than the label homophily of the corresponding static graph.

4.5.3. EVALUATION

METRICS

We evaluate the models using performance matrix $\mathbf{M} \in \mathbb{R}^{T \times T}$, where $\mathbf{M}_{i,k}$ denotes the performance score reported by an evaluation metric (e.g. AUC-ROC, average precision etc.) on task $\mathcal{S}_k$ after the model has been trained over a sequence of tasks from $\mathcal{S}_1$ to $\mathcal{S}_i$. At each time step t, the average performance of the model is measured by the average of the model's performances on task $\mathcal{S}_1$ to task $\mathcal{S}_i$, i.e., the average of the row i in performance matrix $\mathbf{M}$. After the whole task sequence is presented to the model, we report the average performance AP as:

$$AP = \frac{\sum_{i=1}^{T} \mathbf{M}_{T,i}}{T} \tag{4.13}$$

which is the higher, the better.

We use the average forgetting AF score proposed in Lopez-Paz & Ranzato [42]. The forgetting on task $\mathscr{S}_i$ is measured by the performance change on task $\mathscr{S}_i$ after the model is trained on the whole task sequence. Formally, we report the average forgetting AF on all the tasks as:

$$AF = \frac{\sum_{i=1}^{T}(\mathbf{M}_{T,i} - \mathbf{M}_{i,i})}{T-1} \tag{4.14}$$

Note that we here compute a single metric to quantify the incurred forgetting over past tasks when the model is trained for the last task $\mathscr{S}_T$. The summand indicates the performance decrease on some task $\mathscr{S}_i$ after learning on later task $\mathscr{S}_T$.

When the average forgetting is negative, its absolute value indicates the averaged performance decrease on all previous tasks when the model is trained on the last task $\mathscr{S}_T$ in the task sequence.

A positive AF score indicates that the performance on some of the past tasks actually increased after training on task $\mathscr{S}_T$. A positive AF score might be the result of correlation among tasks that the model exploited, thus showing an improvement over past tasks.

Such an observation may be when the tasks from a graph are highly correlated with each other, training on the new task would help further improve the performance on the old tasks.

Overall, we report the AP and AF for each model, and the scores we obtain from the two metrics are interpreted in the following Table 4.2.

	high AF	low AF
high AP	preserves well-rounded knowledge across all the tasks	performs well on the new task, while forgetting about the old tasks
low AP	preserves the knowledge from the old tasks and harms the overall performance indicates the tasks are not correlated, improvements on one task harm the performance on the other tasks	forgets about the old task, and fail to perform well on the new task

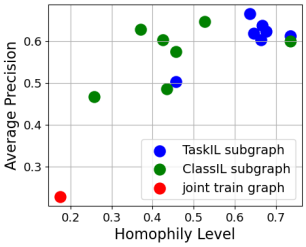
Table 4.2: The interpretation of the average performance score (AP) and the average forgetting score (AF).

VISUALIZATION

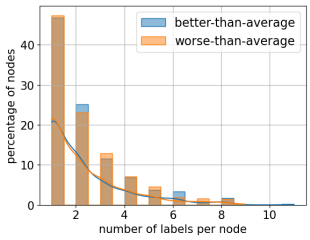
We use the heatmaps and lineplots to visualize the performance matrix $\mathbf{M}$. Due to the limited space, we add the heatmaps in the Appendix 4.8.4. The lineplots are shown in Figure 4.7, which have the time steps as x axis, the y axis indicates the average performance of the model over all the tasks that have been encountered so far.

4.6. RESULTS AND ANALYSIS

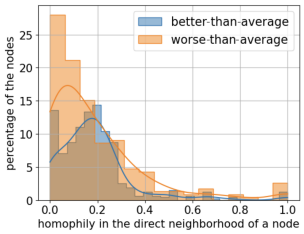
In this section, we summarize the experimental results on the multi-label datasets in the TASK-IL SETTING and CLASS-IL SETTING defined in section 4.2 in the Table 4.3 and Table 4.4, respectively. To use a single numerical value to quantify the overall performance of the



(a) The visualization of the performances of GCN on the subgraphs and the joint train graph from PCG and the label homophily of the graphs.



(b) The distribution of the number of labels per node in the better-than-average subset and in the worse-than-average subset.



(c) The distribution of the label homophily of the nodes in the better-than-average subset and in the worse-than-average subset.

Figure 4.6: Visualization of the analysis on the performance of SIMPLEGCN and JOINTTRAINGCN using PCG as an example.

models, we calculate an average performance matrix $\hat{\mathbf{M}}$ from the performance matrices from the three random splits and report the *AP* and *AF* from the averaged performance matrix.

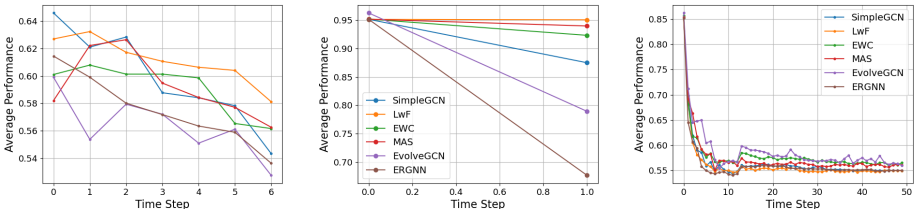
4.6.1. LOWER AND UPPER BOUNDS IN CGL

In the previous CGL frameworks [12, 13], SIMPLEGCN and JOINTTRAINGCN are shown to have the worst and the best performance. Such a result is also expected as (i) SIMPLEGCN is employed on sequential data without any enhanced abilities to deal with catastrophic forgetting (thereby showing performance degradation) and (ii) in JOINTTRAINGCN all data is used to train the base GNN. However, the results from multi-label datasets in both incremental settings, as shown in Table 4.3 and Table 4.4, reveal that SIMPLEGCN and JOINTTRAIN are no longer suitable as lower and upper bounds for evaluating CGL performance in a more generalized scenario of multi-label datasets. In the following, we theoretically and empirically analyze the rationale behind such a finding.

LABEL HOMOPHILY AND GCN

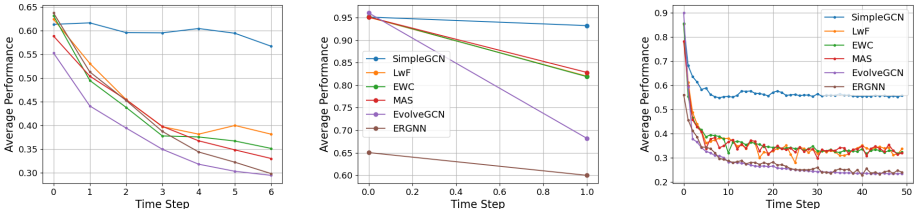
GNNs, specifically GCN, which is used as a base network are known to have better performance on high label-homophilic graphs. As shown in Theorem 2, splitting labels into distinct prediction tasks and creating subgraphs for each task results in an increase in label homophily of the edges in the subgraphs as compared to that in the full graph. In particular, if in a dataset there are a large number of single-labeled nodes in the full graph with a non-zero edge label homophily, the increase in label homophily of edges in subgraphs helps SIMPLEGCN to assign correct labels to the corresponding nodes. However, in JOINTTRAIN, the presence of diverse neighborhoods around single-labeled nodes leads to low label homophily, impacting its performance negatively.

Empirical evidence. Figure 4.6 illustrates the above statements with an example from one random shuffle of PCG using the subgraphs generated for the TASK-IL SETTING (colored in blue), CLASS-IL SETTING (colored in green) and the original static graph given by the dataset (colored in red). On the x axis, we show the label homophily level of the input



(a) PCG **(b)** DBLP **(c)** YELP

In TASK-IL SETTING



(d) PCG **(e)** DBLP **(f)** YELP

In CLASS-IL SETTING

Figure 4.7: Learning curves showing the dynamics of the average performance during learning on the task sequences of different datasets. The color coding and legend names remain consistent across all subfigures. To avoid obstructing the line plot, we omit the legend in the subplots corresponding to PCG.

graphs, while on the y axis, we show the performance of SIMPLEGCN after it is trained on the subgraph in the corresponding incremental settings and JOINTTRAINGCN on the whole static graph. We make the following observations.

- The subgraphs in TASK-IL SETTING and CLASS-IL SETTING have higher label homophily than the full graph, explaining the better performance of SIMPLEGCN as compared to JOINTTRAINGCN.
- We also observe that as compared to TASK-IL SETTING, the subgraphs generated for CLASS-IL SETTING have lower label homophily. This happens because of expanding label sets in CLASS-IL SETTING.

In Figure 4.6b and 4.6c, we further analyze the causes of the bad performance of the JOINTTRAINGCN. We used the JOINTTRAINGCN model on test nodes from the joint train graph in PCG and calculated an average precision score for each node. The mean value of the scores is then used as a threshold to divide the test nodes into the set of nodes that perform *better-than-average* and the *worse-than-average* performing node subset, indicated by the blue and orange bars in the plots. To remove the influence of the difference in the sizes of the subsets, we use the percentage of the nodes in the corresponding subset as the y axis.

Based on the edge homophily defined in 5, we define the label homophily in the direct neighborhood of a node as the averaged edge homophily connected to this node:

Definition 6. *For a node v in the graph $\mathcal{G}$, we define the label homophily of a node v with respect to its immediate neighborhood $\mathcal{N}^v$, represented as h^v, as the average of label homophily of the edges connected to v:*

$$h^v = \frac{\sum_{e(i,j)|j \in \mathcal{N}^v} h^{e(i,j)}}{|\mathcal{N}^v|}$$

We make the following observations.

- In Figure 4.6b, the percentage of single-labeled nodes in the worse-than-average performing subset is higher than that the better-than-average subset.
- Figure 4.6c shows that in fact, the high percentage of worse-performing nodes have very low label homophily (computed using Definition 6) close to 0.
- The above two observations indicate that the performance of JOINTTRAINGCN suffers due to the presence of a higher percentage of low label homophily edges with at least one single-labeled node.

For completeness, we include in Figure 4.6c a Kernel Density Estimation on the node homophily distribution, which shows a clear shift in the distributions of the label homophily in the better-performing subset as compared to the worse-than-average subset.

In the following sections, we summarize the performance of the chosen baselines in the TASK-IL SETTING and CLASS-IL SETTING and provide a detailed analysis of the performances of the baselines on different datasets.

4.6.2. RESULTS IN TASK-IL SETTING

Table 4.3: Performance of the baseline models in the TASK-IL SETTING setting. The performances are reported in Average Precision. "AP" stands for Average Precision, and the higher, the better. "AF" indicates the average forgetting, and the higher, the better.

TASK-IL	PCG		DBLP		YELP	
	AP	AF	AP	AF	AP	AF
SIMPLEGCN	54.34 ± 0.04	-6.11 ± 0.03	87.47 ± 0.12	-15.76 ± 0.00	54.87 ± 0.03	-1.43 ± 0.05
LWF	58.12 ± 0.05	-2.84 ± 0.02	95.01 ± 0.01	-0.98 ± 0.00	54.89 ± 0.03	-2.07 ± 0.05
EWC	56.17 ± 0.03	-3.77 ± 0.03	92.28 ± 0.05	-6.51 ± 0.01	56.53 ± 0.05	-0.17 ± 0.02
MAS	56.26 ± 0.04	-2.64 ± 0.03	93.93 ± 0.03	-3.17 ± 0.00	56.05 ± 0.03	-0.76 ± 0.05
EVOLVEGCN	52.76 ± 0.06	-3.68 ± 0.03	78.94 ± 0.25	-35.20 ± 0.00	55.93 ± 0.07	-5.11 ± 0.07
ERGNN	53.64 ± 0.06	-1.39 ± 0.02	67.70 ± 0.03	-24.96 ± 0.00	54.99 ± 0.03	-0.92 ± 0.04
JOINTTRAIN	22.47 ± 0.47	–	85.60 ± 0.25	–	13.80 ± 0.08	–

Table 4.3 presents results for three real-world multi-label datasets in TASK-IL SETTING. In general, the knowledge distillation method LWF excels on graphs with shorter

task sequences (e.g., PCG and DBLP with 7 and 2 tasks, respectively). In contrast, all methods perform comparably on the graph with a long task sequence in YELP with 50 tasks, among which regularization-based methods like EWC and MAS slightly outperform other approaches. This disparity arises because LWF distills knowledge only from the last time step, leading to a performance drop with longer sequences. Meanwhile, regularization-based methods, like EWC and MAS, which penalize the changes in the important parameters for previous tasks, prove effective for longer task sequences. The weak performance of the replay-based methods ERGNN indicates the importance of including the local topological structure around the nodes in the buffer instead of sampling isolated nodes in the buffer. Dynamic graph neural networks like EVOLVEGCN struggle with substantial forgetting despite achieving notable average precision scores because they only focus on the current task. We visualize the learning curve of the models in the TASK-IL SETTING on PCG, DBLP, and YELP in Figure 4.7a, 4.7b, and 4.7c, respectively. The x axis indicates the current time step, and the corresponding value on the y axis infers the average performance of the model at the current time step over all the tasks encountered so far.

4.6.3. DETAILED ANALYSES ON DIFFERENT DATASETS

PCG. PCG has a relatively shorter task sequence with 7 tasks. SIMPLEGCN showcases competitive scores but is susceptible to forgetting, indicating the low correlation among the tasks. LWF outperforms SIMPLEGCN and notably improved robustness against forgetting, which indicates the shorter task sequence in PCG contributes to the effectiveness of LWF in retaining task knowledge because LWF only distills knowledge from the previous model. EWC and MAS also exhibit competitive performance, demonstrating moderate resistance to forgetting. Meanwhile, because of the low correlations among the tasks, EVOLVEGCN faces challenges using with a lower performance and notable forgetting. JOINTTRAINGCN has the poorest performance because of the low label homophily level on the joint train graph.

DBLP. DBLP has the shortest task sequence length with only 2 tasks. LWF once again stands out with the highest performance and minimal forgetting. The SIMPLEGCN has the worst forgetting on DBLP compared to the other two datasets, indicating the tasks in DBLP have the lowest task correlation. While EWC and MAS present comparable performance to LWF, they suffer from worse forgetting. Notably, the low task correlation also results in the low performance and extreme forgetting of EVOLVEGCN and ERGNN, indicating the information from the previous task that lies in the model, and the data can not assist the model's performance on the new task. And because the joint train graph in DBLP has the highest level of label homophily, the JOINTTRAINGCN also achieves better performances compared to its performance on the other two multi-label datasets.

YELP. The YELP dataset is characterized by the longest task sequence encompassing 50 tasks and featuring the highest task correlations, which is indicated by the competitive performance shown by SIMPLEGCN. Despite the extended task sequence, training on a new task does not significantly impair performance on the previous tasks. The long task sequence poses a potential challenge for LWF, as prolonged sequences lead to increased forgetting. EWC and MAS emerge as robust performers in this demanding setting, demonstrating solid performance with competitive performances and modest forgetting. EVOLVEGCN encounters a lower score coupled with considerable forgetting, as the high task correlation

makes the utilization of the previous model helpful to improve the performance of the current task, EVOLVEGCN pays no attention to maintaining the performance on the old tasks. Additionally, ERGNN achieves a comparable performance with minimal forgetting, positioning it as a strong contender on the YELP dataset. JOINTTRAINGCN achieves the lowest performance because of the low label homophily level in the joint train graph.

4.6.4. CLASS-IL SETTING

Table 4.4: Performance of the baseline models in CLASS-IL SETTING setting. The performances are reported in Average Precision. "AP" stands for Average Precision and the higher the better. "AF" indicates the average forgetting, and the higher, the better.

CLASS-IL	PCG		DBLP		YELP	
	AP	AF	AP	AF	AP	AF
SIMPLEGCN	56.70 ± 0.04	-2.48 ± 0.03	93.22 ± 0.03	-3.90 ± 0.00	55.78 ± 0.03	-1.56 ± 0.05
LWF	38.13 ± 0.13	3.48 ± 0.03	81.98 ± 0.18	-0.69 ± 0.00	33.72 ± 0.05	0.59 ± 0.05
EWC	35.12 ± 0.13	2.17 ± 0.03	81.88 ± 0.06	-8.92 ± 0.00	32.32 ± 0.08	1.71 ± 0.03
MAS	33.00 ± 0.15	0.98 ± 0.02	82.82 ± 0.10	-4.87 ± 0.00	32.07 ± 0.07	-1.10 ± 0.04
EVOLVEGCN	29.44 ± 0.12	0.23 ± 0.01	68.18 ± 0.03	-29.69 ± 0.00	23.45 ± 0.06	-0.70 ± 0.05
ERGNN	29.80 ± 0.14	-2.82 ± 0.03	59.99 ± 0.12	3.14 ± 0.00	24.00 ± 0.06	-0.12 ± 0.01
JOINTTRAIN	22.47 ± 0.47	–	85.60 ± 0.25	–	13.80 ± 0.08	–

Table 4.4 presents results for three real-world multi-label datasets in CLASS-IL SETTING. Overall, SIMPLEGCN achieves a superior performance across all datasets. This performance contrast is noteworthy when compared to its performance on multi-class datasets in the previous works [12, 13]. The key distinction lies in our evaluation framework, where we enable the label vectors of multi-labeled nodes to expand during CLASS-IL SETTING. In essence, this approach incorporates the previous labels of multi-labeled nodes as part of the target labels in subsequent tasks. This strategy serves a dual purpose: it mitigates the problem of forgetting while simultaneously improving the performance on earlier tasks. This improvement is indicated by the positive average forgetting scores in the CLASS-IL SETTING context. The performance of JOINTTRAINGCN is not influenced by the change in the setting, as it is trained on all the tasks simultaneously.

The drop in the performances of other baseline models is a result of the increasing number of classes in the tasks at each time step, i.e., the difficulty of the task increases at each time step. We visualize the learning curve of the models in the CLASS-IL SETTING on PCG, DBLP, and YELP in Figure 4.7d, 4.7e, and 4.7f, respectively. The x axis indicates the current time step, and the corresponding value on the y axis infers the average performance of the model at the current time step over all the tasks encountered so far. Below, we analyze the performance of the chosen baseline models on each of the datasets.

4.6.5. DETAILED ANALYSES ON DIFFERENT DATASETS

PCG. SIMPLEGCN leads with the highest average performance overall tasks with low forgetting, as it has no CL technique to prevent forgetting. On the other hand, the CL methods sacrificed the average performance on the task sequence but successfully maintained a positive AF. This means the knowledge distillation- and regularization-based models are able to retain the knowledge from the old tasks in the CLASS-IL SETTING. EVOLVEGCN and

ERGNN achieve comparable average performance on the task sequence, but the ERGNN fails to retain the knowledge from the old task as it only samples the isolated nodes in the replay buffer while ignoring the topological structure. JOINTTRAINGCN remains the worst-performing model because of the low label homophily in the input graph.

DBLP. DBLP has the shortest task sequence, but SIMPLEGCN and CL methods LWF, EWC, and MAS suffered from the most severe forgetting problem on it. These negative average forgetting scores observed in DBLP indicate low task correlation, i.e., the knowledge from the old task hinders the model from achieving better performance on the new task. As the least multi-labeled graph, DBLP witnesses the least pronounced performance dip in the CLASS-IL SETTING compared to TASK-IL SETTING. This observation suggests that multi-label datasets pose a more challenging test for models when the label vectors of nodes continue to grow.

YELP. In YELP, nodes are more multi-labeled compared to PCG and DBLP, as shown in Table 4.1. Overall, we see a clear performance difference in CLASS-IL SETTING compared to TASK-IL SETTING on YELP. Furthermore, knowledge distillation- and regularization-based methods surpass the dynamic graph neural network EVOLVEGCN and the replay-based method ERGNN. This is primarily due to the fact that EVOLVEGCN neglects the preservation of knowledge from previous tasks, which ultimately hampers overall performance. ERGNN, on the other hand, disregards the topological structure surrounding the sampled experience nodes, further impacting its efficacy in handling the evolving tasks.

4.7. CONCLUSION

We develop a new evaluation framework which we refer to as AGALE for continual graph learning. Filling in the gaps in the current literature, we (i) define two generalized incremental settings for the more general multi-label node classification task, (ii) develop new data split algorithms for curating CGL datasets, and (iii) perform extensive experiments to evaluate and compare the performance of methods from continual learning, dynamic graph learning and continual graph learning. Through our theoretical and empirical analyses we show important differences of the multi-label case with respect to the more studied single-label scenario. We believe that our work will encourage the development of new methods tackling the general scenario of multi-label classification in continual graph learning.

Following the current literature, we focus on quantifying catastrophic forgetting in AGALE. In realistic scenarios, there is also the case where the model could be required to selectively forget about the past. For example, users in the social network might not further show interest in certain topics and un-follow some of the friends. Developing new evaluation metrics as well as new models to reward selective forgetting of some tasks while avoiding catastrophic forgetting overall is an interesting avenue for future research.

4.8. APPENDIX

Organization. We analyze the characteristics of the subgraphs generated by AGALE and compare them with the full graph given in the dataset in Section 4.8.1. Furthermore, we also apply our AGALE on single-label graph CORAFULL and summarize and analyze the results in section 4.8.2 to further demonstrate the generalization of AGALE in sing-label scenarios.

Additionally, we provide detailed time and space complexity analysis in Section 4.8.3 and measure and summarize the run time of the conducted experiments as well. Last but not least, we provide the visualization of the performance matrix using heatmaps in Section 4.8.4.

4.8.1. DATA ANALYSIS OF THE SUBGRAPHS

In this section, we present an analysis of the subgraphs derived by our evaluation framework from the static graph in PCG, showcasing the efficacy of our approach. Figure 4.8 illustrates the degree distribution of nodes within the seven subgraphs generated from the PCG dataset. We see from the degree distribution that the nodes in the subgraphs also have a similar degree distribution to the nodes in the original static graph.

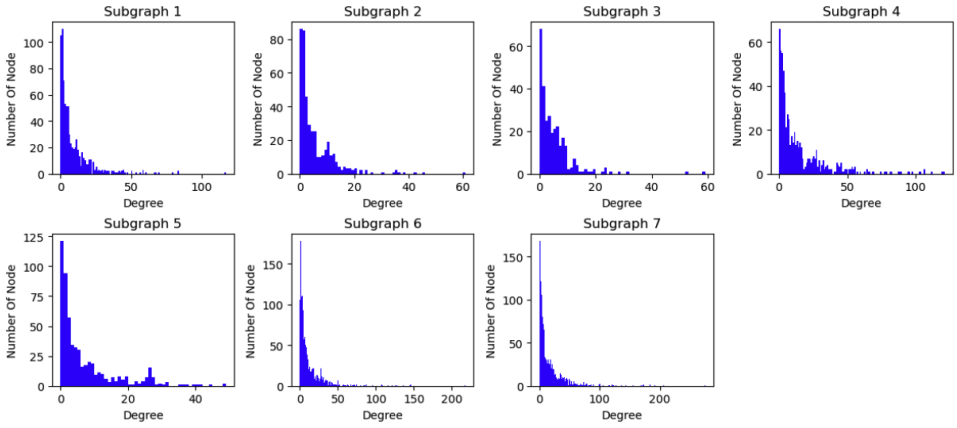


Figure 4.8: The Node Degree Distribution In the seven Subgraphs Generated From PCG.

4.8.2. APPLICATION OF OUR EVALUATION FRAMEWORK ON SINGLE-LABEL GRAPHS

In this section, we provide an example of applying our evaluation framework to single-label graphs. Here, we use CORAFULL as an example. We summarize the characteristics of CORAFULL in Table 4.5. As shown in the Table, CORAFULL has 70 classes, which are divided into 35 tasks in 3 random orders.

In Table 4.6 and 4.7, we summarize the performance of LWF and ERGNN on the dataset CORAFULL in TASK-IL SETTING and CLASS-IL SETTING and use the line plots in Figure 4.9 to visualize the learning curves of the chosen models in the two settings on CORAFULL.

4.8.3. TIME AND SPACE COMPLEXITY ANALYSIS

Here, we provide theoretical time and space complexity analysis of the models used in this work.

Table 4.5: The data statistics. Specifically, $|\mathcal{V}|$, $|\mathcal{E}|$, $|\mathcal{C}|$, $\overline{|\mathcal{L}|}$, and r_{homo} denote the number of nodes, edges, classes, mean label count per node, and label homophily of the static graph given by the dataset, respectively. $|T|$ signifies the count of tasks in the resulting task sequence. Additionally, $\overline{|\mathcal{V}|}$ and $\overline{|\mathcal{E}|}$ represent the average number of nodes and edges in a subgraph. Further details on label homophily are captured through $\overline{|r|}_{tsk}$ and $\overline{|r|}_{cls}$, representing the averaged label homophily of subgraphs in the TASK-IL SETTING and CLASS-IL SETTING), respectively.

	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{C}\|$	$\|T\|$	r_{homo}	$\overline{\|\mathcal{V}\|}$	$\overline{\|\mathcal{E}\|}$	$\overline{\|r\|}_{tsk}$	$\overline{\|r\|}_{cls}$
CORAFULL	$19K$	$130K$	70	35	0.57	566	1035	0.99	0.99

Table 4.6: Performance of the baseline models in TASK-IL SETTING setting. The performances are reported in Average Precision. "AP" stands for Average Precision and the higher the better. "AF" indicates the average forgetting, and the higher, the better.

TASK-IL SETTING	CORAFULL	
	AP	AF
LWF	53.46 ± 0.12	-9.53 ± 0.16
ERGNN	59.49 ± 0.20	4.37 ± 0.34

Table 4.7: Performance of the baseline models in CLASS-IL SETTING setting. The performances are reported in Average Precision. "AP" stands for Average Precision and the higher the better. "AF" indicates the average forgetting, and the higher, the better.

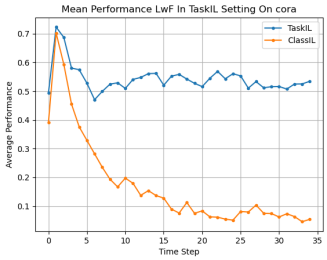
CLASS-IL SETTING	CORAFULL	
	AP	AF
LWF	5.42 ± 0.15	-7.45 ± 0.14
ERGNN	40.39 ± 0.27	-56.08 ± 0.25

Complexity analysis for the base model. As the base model used by all compared methods is GCN [39], we first analyze its complexity. To keep the notations simpler let us assume that the feature (including the input features) dimension in all layers is equal to d. Let n, m denote the number of nodes and edges, respectively in the input graph at any time point. For the sake of brevity in the presentation, we assume that the number of nodes and edges stay the same for all time points.

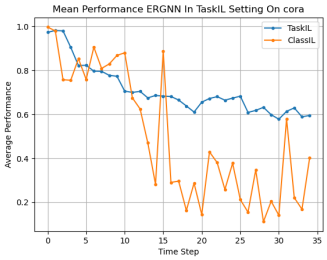
For GCN, at each layer, the operation includes feature transformation, neighbourhood aggregation, and activation. The feature transformation over two layers leads to the multiplication of matrices of sizes (i) $n \times d$ and $d \times d$, and (ii) $d \times d$ and $d \times d$ which leads to a total time complexity of $\mathcal{O}(nd^2)$.

And the neighborhood aggregation requires a multiplication between matrices of size $n \times n$ and $n \times d$, yielding $\mathcal{O}(n^2 d)$. In practice, we compute this using a sparse operator, such as the PyTorch scatter function for each entry (i, j) in the adjacency matrix of the edge $e \in \mathcal{E}$, which yields a total cost of $\mathcal{O}(md)$. Finally, the activation is an element-wise function with the time complexity of $\mathcal{O}(n)$. Overall, the time complexity of a L layer GCN is $\mathcal{O}(nd^2L + mdL + nL)$.

For computing space requirements of GCN, we include (i) the space required for the



(a) LWF in TASK-IL SETTING and CLASS-IL SETTING on CORAFULL



(b) ERGNN in TASK-IL SETTING and CLASS-IL SETTING on CORAFULL

Figure 4.9: Our Framework on Single-label Datasets

input adjacency matrix of size $n \times n$, (ii) the feature matrix of size $n \times d$, and (iii) the model itself with $d^2 + d$ parameters for weight and bias in each layer. In total, the space complexity of GCN is $\mathcal{O}(n^2 + nd + L(d^2 + d))$.

As all methods mentioned in this work either use GCN as the backbone model or are built upon GCN, we denote in the following discussion and the Table 4.8 the time and space requirement of GCN as T_{GCN} and S_{GCN} respectively.

Complexity analysis of SIMPLEGCN. SIMPLEGCN trains a GCN for each time step $t \in \mathcal{T}$. And because it does not apply any continual learning techniques to remember from previous time steps, the time is the same with GCN, i.e., $\mathcal{O}(|\mathcal{T}|T_{GCN})$ and the space complexity is S_{GCN}.

Complexity analysis of LWF. LWF uses GCN as the backbone model, the GCN is trained at each time step $t \in \mathcal{T}$ for the new task, which gives the time complexity of $\mathcal{O}(|\mathcal{T}|T_{GCN})$. To do the knowledge distillation, the previous model also calls GCN forward passes with time complexity of $\mathcal{O}(|\mathcal{T}|T_{GCN})$. Overall, the time complexity is $\mathcal{O}(2|\mathcal{T}|T_{GCN})$. The space consumption consists of loading the current GCN model and the previous GCN model for the knowledge distillation, with space complexity of $\mathcal{O}(2S_{GCN})$.

Complexity analysis of EWC. EWC calls forward passes of GCN at each time step, and for each parameter at each time step $t \in \mathcal{T}$, the values on the diagonal of the fisher matrix are approximated using the value of each model parameter itself and its gradient. This is an element-wise calculation, which gives the time complexity of $\mathcal{O}(|\mathcal{T}| \times M)$, M indicates the number of parameters in the GCN, which is $L(d^2 + d)$. Overall, the time complexity of EWC is the time complexity of GCN plus the calculation of the fisher matrix, which is $\mathcal{O}(|\mathcal{T}| \times (T_{GCN} + M))$. The space requirement of EWC consists of the space requirement of GCN, and the matrix stores the gradients of the parameters at each time step of size $\mathcal{O}(|\mathcal{T}| \times M)$. In total, the space complexity of EWC is $\mathcal{O}(S_{GCN} + |\mathcal{T}|M)$.

Complexity analysis of MAS. Similarly, MAS using GCN as backbone model, at each time step $t \in \mathcal{T}$, there are forward passes of GCN and the calculation of fisher matrix for

parameters, which gives the overall time complexity of $\mathcal{O}(|\mathcal{T}|(T_{GCN}+M))$. The space requirement of MAS consists of the space complexity of GCN and one matrix for the gradient of the parameters of size $\mathcal{O}(M)$, which yields in total $\mathcal{O}(S_{GCN}+M)$.

Complexity analysis of EVOLVEGCN. EVOLVEGCN is a method from the category of Dynamic Graph Neural networks, which trains a new model at each time step with time complexity same as GCN, i.e., T_{GCN} and updates the model parameter using a recurrent neural network using the corresponding parameter from previous time step as input, which has the time complexity of $\mathcal{O}(|\mathcal{T}|nd)$. Overall, EVOLVEGCN is more expensive than the other Continual Learning methods with time complexity of $\mathcal{O}(T_{GCN}+|\mathcal{T}|nd)$. The space requirement of EVOLVEGCN consists of the space requirement of the GCN plus the recurrent unit for the reset, update, and new gate with $3(d^2+d)$. In our implementation, we use one recurrent layer for each layer in GCN. Thus the overall space complexity yields $\mathcal{O}(2S_{GCN}+3M)$.

Complexity analysis of ERGNN. ERGNN is a replay-based method. At each time step $t \in \mathcal{T}$, it retrains the GCN on the current graph and the buffer-nodes-formed graph, and the sampling process goes through the nodes in the new graph, which gives the time complexity of $\mathcal{O}(2|\mathcal{T}|T_{GCN}+n)$. The space requirement of ERGNN consists of the buffer size of $|\mathcal{B}|$ and the space complexity of GCN. In total, it yields the space complexity of $\mathcal{O}(S_{GCN}+|\mathcal{B}|)$.

Complexity analysis of JOINTTRAINGCN. JOINTTRAINGCN has the same time and space complexity as the base model GCN. It uses the whole static graph as input and is only trained once without the task sequence.

The above time and space complexity analyses are summarized in Table 4.8.

Table 4.8: The simplified time complexity analysis. T_{GCN} and S_{GCN} corresponds to the time and space requirement of the base GCN model.

Model	Time Complexity	Space Complexity
SIMPLEGCN	$\mathcal{O}(\|\mathcal{T}\|T_{GCN})$	S_{GCN}
LWF	$\mathcal{O}(2\|\mathcal{T}\|T_{GCN})$	$\mathcal{O}(2S_{GCN})$
EWC	$\mathcal{O}(\|\mathcal{T}\| \times (T_{GCN}+M))$	$\mathcal{O}(S_{GCN}+\|\mathcal{T}\|M)$
MAS	$\mathcal{O}(\|\mathcal{T}\|(T_{GCN}+M))$	$\mathcal{O}(S_{GCN}+M)$
EVOLVEGCN	$\mathcal{O}(T_{GCN}+\|\mathcal{T}\|nd)$	$\mathcal{O}(2S_{GCN}+3M)$
ERGNN	$\mathcal{O}(2\|\mathcal{T}\|T_{GCN}+n)$	$\mathcal{O}(S_{GCN}+\|\mathcal{B}\|)$
JOINTTRAINGCN	T_{GCN}	S_{GCN}

Besides, we also measured the run time of the experiments in this work. The results are summarized in Table 4.9. Note that the running time of the experiments can be biased due to different splits and how the resources are distributed on the computer. The theoretical analysis may provide more insights into the complexity of time.

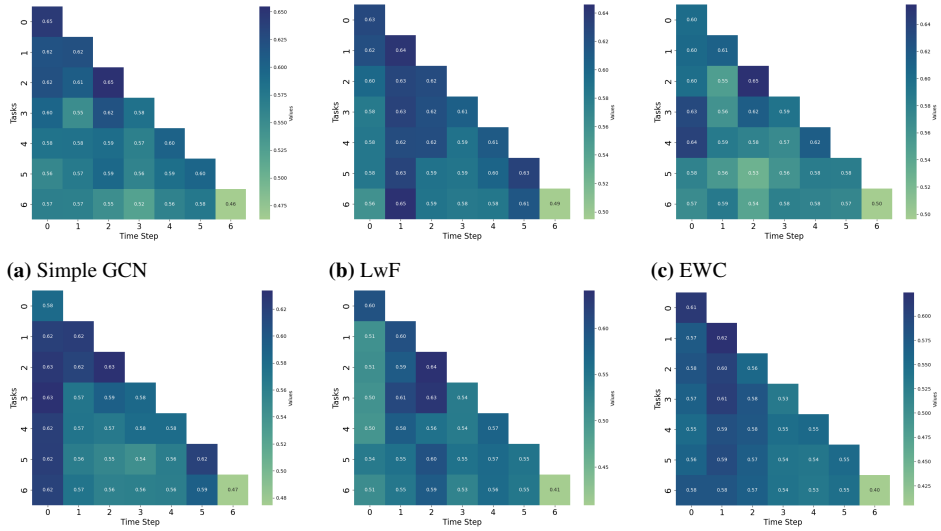
Table 4.9: The computation time of the experiments from Section 4.6 in second. The computation time is measured with one random split for each dataset.

	TASK-IL SETTING			CLASS-IL SETTING		
	PCG	DBLP	YELP	PCG	DBLP	YELP
SIMPLEGCN	43.47	801.85	77163.32	88.31	886.79	104869.58
LWF	49.17	1193.40	142468.01	111.23	732.76	264657.47
EWC	51.32	939.79	79804.48	135.44	790.73	200649.31
MAS	148.19	1169.50	75255.31	135.93	1230.88	145917.67
EVOLVEGCN	40.34	427.99	120580.88	94.94	497.80	310540.41
ERGNN	47.27	536.20	416090.74	172.05	131.57	167624.39
JOINTTRAINGCN	166.82	796.19	80827.72	166.82	796.19	80827.72

4.8.4. VISUALIZATION OF THE PERFORMANCE MATRIX

In this section, we provide the visualization of the performance matrix using the heatmap on the three multi-label datasets. In the heatmap, each cell corresponds to a unique entry in **M**, and its position in the heatmap mirrors its position in the matrix. We use the gradient of the color to indicate the performance. The color intensity indicates the magnitude of the value.

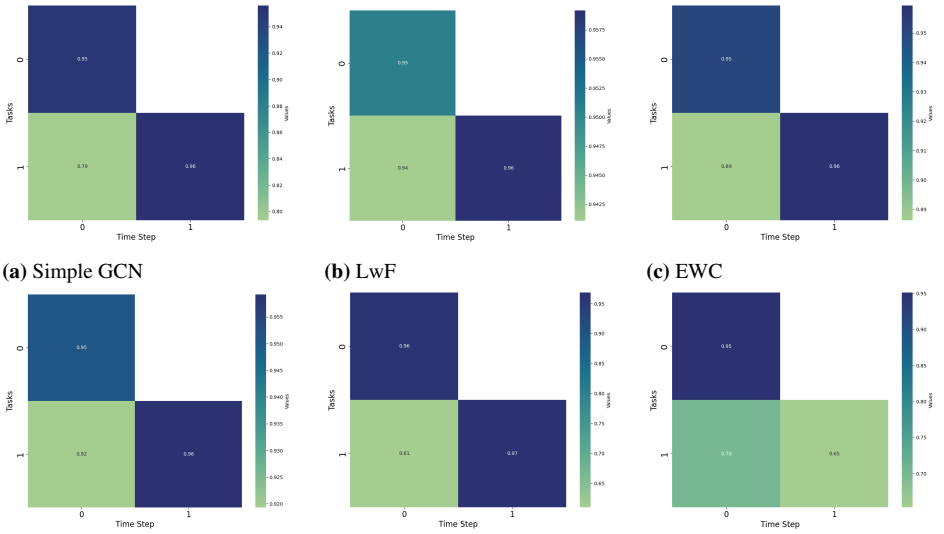
In the Figure 4.10, 4.11, and 4.12, we show the heatmaps correspond to the performance matrices of the baseline models in the TASK-IL SETTING on datasets PCG, DBLP, and YELP, respectively, while in the Figure 4.13, 4.14, and 4.15, we show the heatmaps correspond to the performance matrices of the baseline models in the CLASS-IL SETTING on datasets PCG, DBLP, and YELP, respectively.



(a) Simple GCN **(b)** LwF **(c)** EWC

(d) MAS **(e)** EvolveGCN **(f)** ERGNN

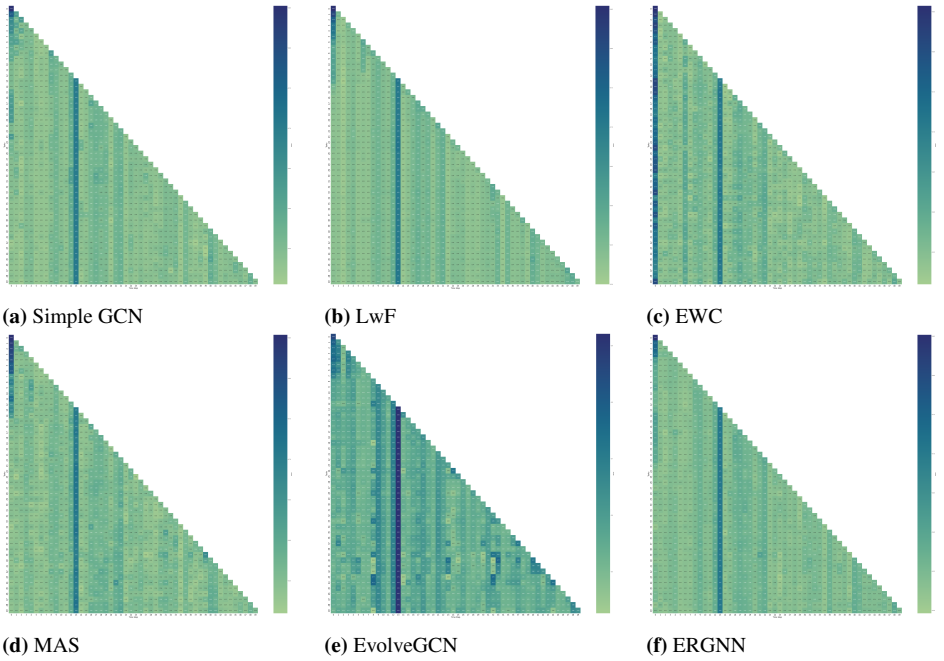
Figure 4.10: Visualization of the performance matrix of the methods in TaskIL setting on dataset PCG



(a) Simple GCN **(b)** LwF **(c)** EWC

(d) MAS **(e)** EvolveGCN **(f)** ERGNN

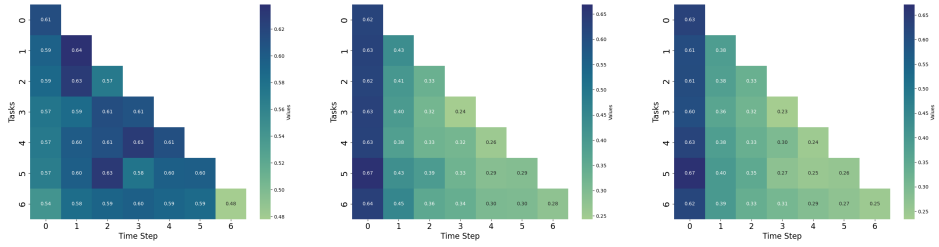
Figure 4.11: Visualization of the performance matrix of the methods in TaskIL setting on dataset DBLP



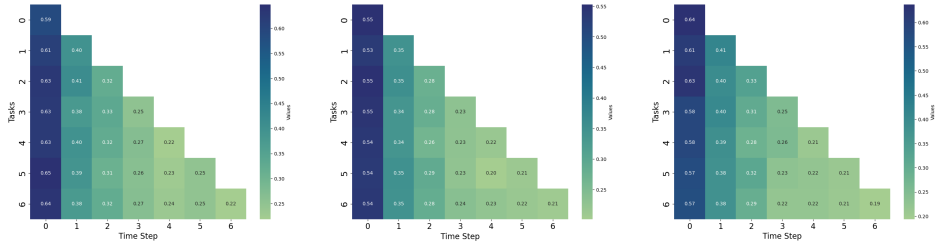
(a) Simple GCN **(b)** LwF **(c)** EWC

(d) MAS **(e)** EvolveGCN **(f)** ERGNN

Figure 4.12: Visualization of the performance matrix of the methods in TaskIL setting on dataset Yelp

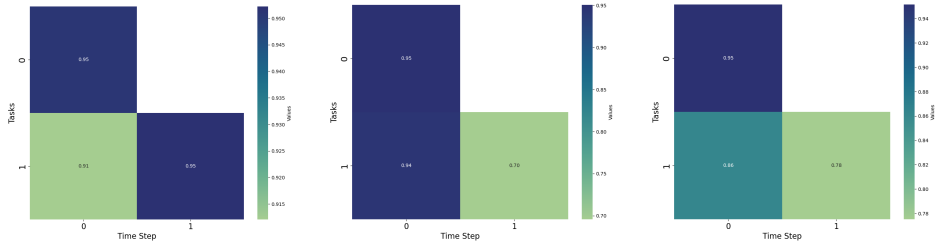


(a) Simple GCN **(b)** LwF **(c)** EWC

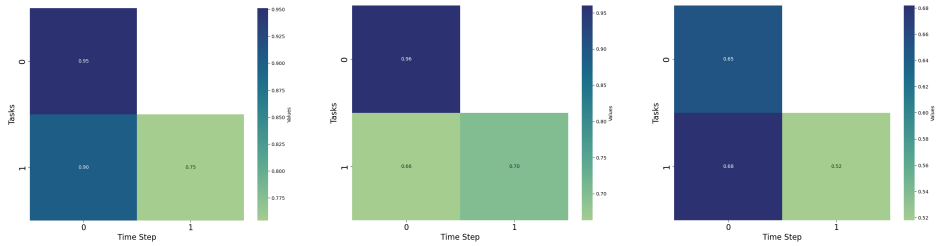


(d) MAS **(e)** EvolveGCN **(f)** ERGNN

Figure 4.13: Visualization of the performance matrix of the methods in CLASS-IL SETTING on dataset PCG

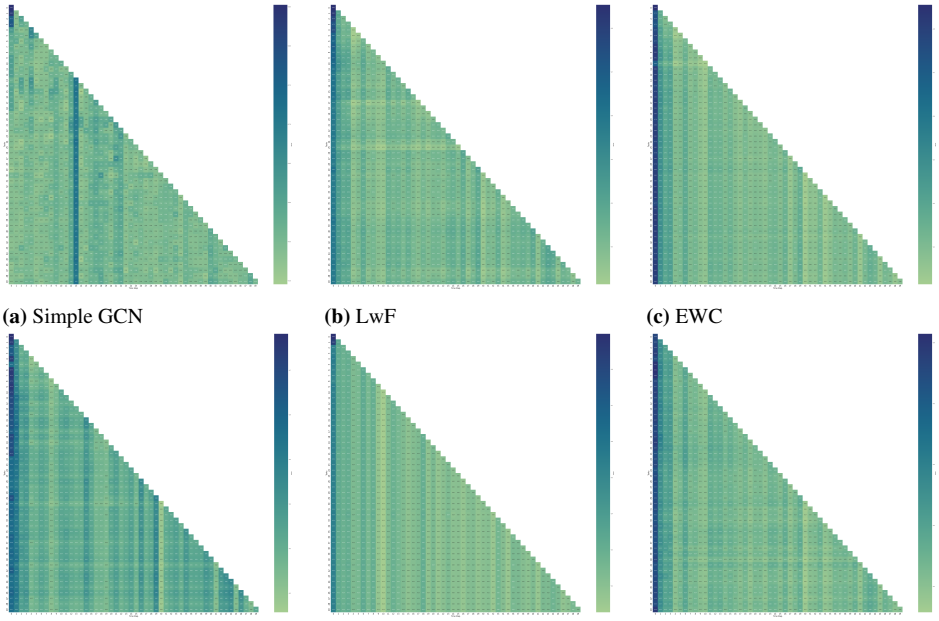


(a) Simple GCN **(b)** LwF **(c)** EWC



(d) MAS **(e)** EvolveGCN **(f)** ERGNN

Figure 4.14: Visualization of the performance matrix of the methods in CLASS-IL SETTING on dataset DBLP



(a) Simple GCN **(b)** LwF **(c)** EWC

(d) MAS **(e)** EvolveGCN **(f)** ERGNN

Figure 4.15: Visualization of the performance matrix of the methods in CLASS-IL SETTING on dataset Yelp

REFERENCES

1. Aljundi, R., Kelchtermans, K. & Tuytelaars, T. Task-Free Continual Learning. *CoRR* **abs/1812.03596.** arXiv: 1812.03596. http://arxiv.org/abs/1812.03596 (2018).
2. Parisi, G. I., Kemker, R., Part, J. L., Kanan, C. & Wermter, S. Continual Lifelong Learning with Neural Networks: A Review. *CoRR* **abs/1802.07569.** arXiv: 1802.07569. http://arxiv.org/abs/1802.07569 (2018).
3. Tang, B. & Matteson, D. S. *Graph-Based Continual Learning* 2021. arXiv: 2007.04813 [stat.ML].
4. Hadsell, R., Rao, D., Rusu, A. A. & Pascanu, R. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences* **24,** 1028–1040 (2020).
5. Van de Ven, G. M. & Tolias, A. S. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734* (2019).
6. Wang, Z., Liu, L. & Tao, D. *Deep Streaming Label Learning* in *Proceedings of the 37th International Conference on Machine Learning* (eds III, H. D. & Singh, A.) **119** (PMLR, 13–18 Jul 2020), 9963–9972. https://proceedings.mlr.press/v119/wang20n.html.
7. Wang, Y., Wang, Z., Lin, Y., Khan, L. & Li, D. *CIFDM: Continual and Interactive Feature Distillation for Multi-Label Stream Learning* in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Association for Computing Machinery, Virtual Event, Canada, 2021), 2121–2125. ISBN: 978-1-4503-8037-9. https://doi.org/10.1145/3404835.3463096.
8. Liu, J., Lin, Y., Ding, W., Zhang, H. & Du, J. Fuzzy Mutual Information-Based Multi-Label Feature Selection With Label Dependency and Streaming Labels. *IEEE Transactions on Fuzzy Systems* **31,** 1–15 (Jan. 2022).
9. Wei, T., Shi, J.-X. & Li, Y.-F. *Probabilistic Label Tree for Streaming Multi-Label Learning* in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Association for Computing Machinery, Virtual Event, Singapore, 2021), 1801–1811. ISBN: 978-1-4503-8332-5. https://doi.org/10.1145/3447548.3467226.
10. Farquhar, S. & Gal, Y. *Towards Robust Evaluations of Continual Learning* 2019. arXiv: 1805.09733 [stat.ML].
11. Lange, M. D., van de Ven, G. & Tuytelaars, T. *Continual evaluation for lifelong learning: Identifying the stability gap* 2023. arXiv: 2205.13452 [cs.LG].
12. Ko, J., Kang, S. & Shin, K. BeGin: Extensive Benchmark Scenarios and An Easy-to-use Framework for Graph Continual Learning. *arXiv preprint arXiv:2211.14568* (2022).
13. Zhang, X., Song, D. & Tao, D. *CGLB: Benchmark Tasks for Continual Graph Learning* in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track* (2022).
14. Kim, C. D., Jeong, J. & Kim, G. Imbalanced Continual Learning with Partitioning Reservoir Sampling. *CoRR* **abs/2009.03632.** arXiv: 2009.03632. https://arxiv.org/abs/2009.03632 (2020).

15. Ma, Y., Liu, X., Shah, N. & Tang, J. Is Homophily a Necessity for Graph Neural Networks? *CoRR* **abs/2106.06134.** arXiv: 2106.06134. https://arxiv.org/abs/2106.06134 (2021).

16. Zhao, T., Dong, T. N., Hanjalic, A. & Khosla, M. Multi-label Node Classification On Graph-Structured Data. *Transactions on Machine Learning Research.* ISSN: 2835-8856. https://openreview.net/forum?id=EZhkV2BjDP (2023).

17. Van de Ven, G. M. & Tolias, A. S. Three scenarios for continual learning. *CoRR* **abs/1904.07734.** arXiv: 1904.07734. http://arxiv.org/abs/1904.07734 (2019).

18. Nguyen, C. V., Li, Y., Bui, T. D. & Turner, R. E. *Variational Continual Learning* 2018. arXiv: 1710.10628 [stat.ML].

19. Aljundi, R., Kelchtermans, K. & Tuytelaars, T. *Task-Free Continual Learning* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019).

20. Li, Z. & Hoiem, D. Learning without Forgetting. *CoRR* **abs/1606.09282.** arXiv: 1606.09282. http://arxiv.org/abs/1606.09282 (2016).

21. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M. & Tuytelaars, T. Memory Aware Synapses: Learning what (not) to forget. *CoRR* **abs/1711.09601.** arXiv: 1711.09601. http://arxiv.org/abs/1711.09601 (2017).

22. Wang, L., Zhang, X., Su, H. & Zhu, J. *A Comprehensive Survey of Continual Learning: Theory, Method and Application* 2023. arXiv: 2302.00487 [cs.LG].

23. Shin, H., Lee, J. K., Kim, J. & Kim, J. Continual learning with deep generative replay. *Advances in neural information processing systems* **30** (2017).

24. Kim, C. D., Jeong, J. & Kim, G. *Imbalanced Continual Learning with Partitioning Reservoir Sampling* 2020. arXiv: 2009.03632 [cs.LG].

25. Lee, J., Yoon, J., Yang, E. & Hwang, S. J. Lifelong Learning with Dynamically Expandable Networks. *CoRR* **abs/1708.01547.** arXiv: 1708.01547. http://arxiv.org/abs/1708.01547 (2017).

26. Febrinanto, F. G., Xia, F., Moore, K., Thapa, C. & Aggarwal, C. *Graph Lifelong Learning: A Survey* 2022. arXiv: 2202.10688 [cs.LG].

27. Yuan, Q. *et al. Continual Graph Learning: A Survey* 2023. arXiv: 2301.12230 [cs.LG].

28. Xu, Y. *et al.* GraphSAIL: Graph Structure Aware Incremental Learning for Recommender Systems. *CoRR* **abs/2008.13517.** arXiv: 2008.13517. https://arxiv.org/abs/2008.13517 (2020).

29. Liu, H., Yang, Y. & Wang, X. Overcoming Catastrophic Forgetting in Graph Neural Networks. *CoRR* **abs/2012.06002.** arXiv: 2012.06002. https://arxiv.org/abs/2012.06002 (2020).

30. Zhou, F. & Cao, C. Overcoming Catastrophic Forgetting in Graph Neural Networks with Experience Replay. *Proceedings of the AAAI Conference on Artificial Intelligence* **35,** 4714–4722. https://ojs.aaai.org/index.php/AAAI/article/view/16602 (May 2021).

31. Wang, J., Song, G., Wu, Y. & Wang, L. Streaming Graph Neural Networks via Continual Learning. *CoRR* **abs/2009.10951.** arXiv: 2009.10951. https://arxiv.org/abs/2009.10951 (2020).

32. Wang, T., Luo, D., Cheng, W., Chen, H. & Zhang, X. *DyExplainer: Explainable Dynamic Graph Neural Networks* 2023. arXiv: 2310.16375 [cs.LG].

33. Yu, W. *et al. NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks* in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Association for Computing Machinery, London, United Kingdom, 2018), 2672–2681. ISBN: 978-1-4503-5552-0. https://doi.org/10.1145/3219819.3220024.

34. Yu, W., Cheng, W., Aggarwal, C. C., Chen, H. & Wang, W. *Link Prediction with Spatial and Temporal Consistency in Dynamic Networks* in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17* (2017), 3343–3349. https://doi.org/10.24963/ijcai.2017/467.

35. Xu, D., Cheng, W., Luo, D., Liu, X. & Zhang, X. *Spatio-Temporal Attentive RNN for Node Classification in Temporal Attributed Graphs* in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19* (International Joint Conferences on Artificial Intelligence Organization, July 2019), 3947–3953. https://doi.org/10.24963/ijcai.2019/548.

36. Rossi, E. *et al.* Temporal Graph Networks for Deep Learning on Dynamic Graphs. *CoRR* **abs/2006.10637.** arXiv: 2006.10637. https://arxiv.org/abs/2006.10637 (2020).

37. Pareja, A. *et al.* EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. *CoRR* **abs/1902.10191.** arXiv: 1902.10191. http://arxiv.org/abs/1902.10191 (2019).

38. Liu, J., Lin, Y., Ding, W., Zhang, H. & Du, J. Fuzzy Mutual Information-Based Multilabel Feature Selection With Label Dependency and Streaming Labels. *IEEE Transactions on Fuzzy Systems* **31,** 77–91 (2023).

39. Kipf, T. N. & Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. arXiv: 1609.02907. http://arxiv.org/abs/1609.02907 (2016).

40. Akujuobi, U., Han, Y., Zhang, Q. & Zhang, X. Collaborative Graph Walk for Semi-supervised Multi-Label Node Classification. *CoRR* **abs/1910.09706.** arXiv: 1910.09706. http://arxiv.org/abs/1910.09706 (2019).

41. Zeng, H., Zhou, H., Srivastava, A., Kannan, R. & Prasanna, V. K. GraphSAINT: Graph Sampling Based Inductive Learning Method. *CoRR* **abs/1907.04931.** arXiv: 1907.04931. http://arxiv.org/abs/1907.04931 (2019).

42. Lopez-Paz, D. & Ranzato, M. Gradient Episodic Memory for Continuum Learning. *CoRR* **abs/1706.08840.** arXiv: 1706.08840. http://arxiv.org/abs/1706.08840 (2017).

5

NodePro: An Instance-Level Profiling Framework for Graph-Structured Data

[1] This chapter is based on the manuscript: **Tianqi Zhao**, Russa Biswas, Megha Khosla, *NodePro: An Instance-Level Profiling Framework for Graph-Structured Data*, currently under review for the *Journal of Data-centric Machine Learning Research (DMLR)*.

5.1. INTRODUCTION

Graph machine learning models have achieved remarkable success on node classification tasks. However, their performance is far from uniform across all nodes. Existing evaluation practices remain largely coarse-grained, relying on aggregate metrics such as accuracy or F1 score. While these metrics are valuable for benchmarking, they obscure important variation in how models behave at the node level. In particular, they fail to reveal why certain nodes are consistently harder to classify whether due to feature sparsity, structural ambiguity, or label noise.

Crucially, models trained on the same dataset and achieving similar overall accuracy may still generalize in fundamentally different ways—failing on different subsets of nodes or exhibiting varying levels of predictive uncertainty. Without a principled approach to capturing node-level learning difficulty and failure modes, we lack the tools to assess the trustworthiness of individual predictions.

Bridging this gap between high-level evaluation and fine-grained model behavior is essential for developing robust, interpretable, and trustworthy graph learning systems. This limitation motivates our central research question: *How can we characterize fine-grained differences in model behavior across individual nodes, beyond aggregate performance metrics?*

To address this question, we propose GNN-MULTIFIX, a node profiling framework designed to characterize individual nodes and explain model behavior at a fine-grained level. Rather than focusing on improving model performance through data augmentation or architectural changes, GNN-MULTIFIX seeks to uncover where models fail by identifying nodes that are consistently misclassified and analyzing the underlying factors that contribute to their difficulty. In doing so, our approach introduces a novel contribution to the growing field of data-centric graph machine learning [1, 2], which has traditionally emphasized improving datasets to enhance model performance. GNN-MULTIFIX instead shifts the emphasis toward understanding model behavior in relation to intrinsic node characteristics, offering a complementary path toward building more robust and interpretable graph learning systems.

GNN-MULTIFIX profiles nodes along two complementary dimensions: data-centric characteristics, which describe a node's context in the input graph, and model-centric behavior, which captures prediction dynamics during training. The data-centric component constructs interpretable node profiles based on three key properties: (i) feature dissimilarity (how distinct a node's features are relative to others in the same class), (ii) local label uncertainty (the diversity of class labels in its immediate neighborhood), and (iii) higher-order structural ambiguity (how consistently distant neighbors share the node's label). Together, these properties summarize the intrinsic difficulty of a node from a data perspective.

To understand how models behave in relation to these profiles, GNN-MULTIFIX analyzes two additional signals: prediction consistency indicating how stable predictions are across training checkpoints, and prediction confidence which quantifies how far the predicted probability for the true label deviates from a random guess. By aligning these model-level signals with data-centric node profiles, GNN-MULTIFIX provides a powerful lens into model generalization and reliability.

We demonstrate that GNN-MULTIFIX not only explains node-level performance but also enables new capabilities such as detecting potentially incorrect predictions and flagging

atypical or anomalous nodes, particularly in noisy or structurally complex graphs where traditional evaluation metrics fall short.

5.2. RELATED WORK

5.2.1. DATA-CENTRIC GRAPH MACHINE LEARNING(DC-GML)

Methods in DC-GML aim to improve model performance by focusing on the quality and manipulation of graph data itself, rather than solely on model architecture [1]. Recent research categorizes DC-GML into stages such as data collection, improvement, and maintenance, emphasizing interventions on graph topology, features, and labels to combat noise, missing information, and mislabeling [1, 3].

Key strategies include augmenting node features to handle incompleteness, refining the labels, and using graph condensation or synthesis to improve the efficiency of the models [1, 4, 5]. These techniques have proven to be efficient to improve the performance of the existing SOTA models on tasks like GNN calibration, anomaly detection, and discovering hidden graph hierarchies.

Within the DC-GML context, graph anomaly detection introduces taxonomies to classify nodes or entire graphs as hard/easy, typical/atypical, in distribution (ID) or OOD [6–8]. These methods are often applied to domain-specific datasets in bio-informatics, social networks, and finance. The goal is to make OOD representations more distinguishable from ID ones. Detection techniques include density-based heuristics [9], distance-based methods such as K-nearest neighbors [10], and learnable scoring strategies that increase the separation between ID and OOD during training [11]. While most approaches focus on graph-level detection, [7] propose a diffusion-based graph generator to synthesize node-level training data, which can be directly integrated into existing GNN models to enhance performance.

Despite its promise, DC-GML still lacks frameworks for understanding how individual nodes affect model learning process. Most existing methods focus on global data properties or modifications, overlooking instance-level behavior.

5.2.2. DATA HARDNESS IN DATA-CENTRIC AI

Data hardness in data-centric AI refers to identifying and managing difficult samples within datasets that impede model learning, particularly in image, tabular and visual tasks [12, 13]. These hard samples include mislabeled, ambiguous, or rare examples that challenge standard models. Addressing data hardness involves techniques like data augmentation to diversify training data and improve model robustness.

Hardness Characterization Methods are algorithmic tools used to detect such challenging samples. Recent research emphasizes evaluation of the data hardness quantitatively across benchmarks such as CIFAR and MNIST, aiming for reliable hardness detection and improved generalization of the models [12]. Managing data hardness is critical for improving model accuracy, robustness, and trustworthiness in data-centric AI [12, 14].

In node-level tasks on graphs, hardness is influenced not just by the features of a node, as in Euclidean data, but also by its surrounding topology, from local structure to higher-order neighborhoods. This added complexity calls for a holistic, graph-aware profiling approach that captures both node attributes and relational context.

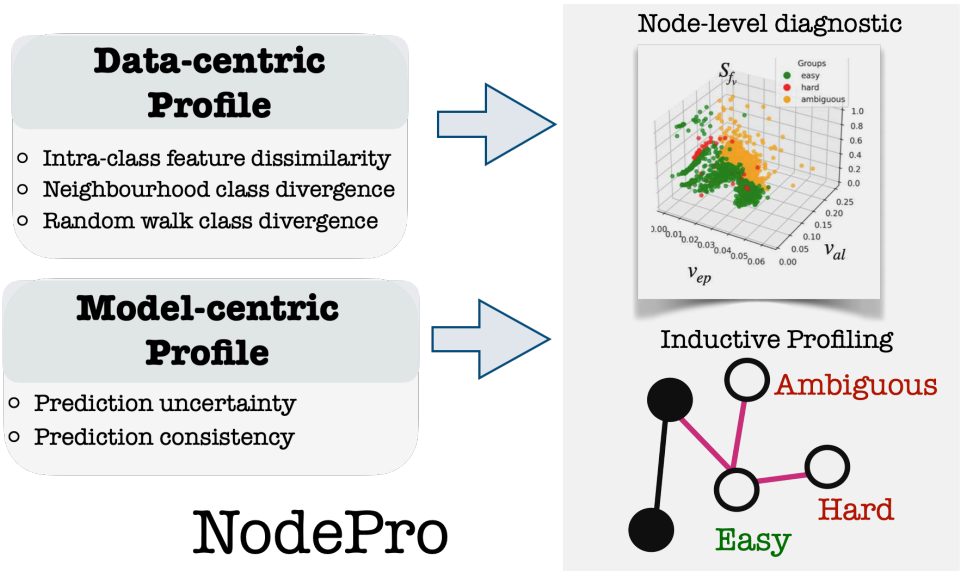


Figure 5.1: GNN-MultiFix combines data- and model-centric signals to diagnose node-level model behavior, enabling fine-grained model comparison, prediction reliability estimation, and error detection.

5.3. Our Proposed Framework

We introduce GNN-MultiFix (see Figure 5.1), a node profiling framework designed to provide fine-grained insights into model behavior on graph-structured data. GNN-MultiFix consists of three main components: (i) data-centric node profiling that captures the intrinsic difficulty of nodes based on their features and structural context, (ii) model-centric node profiling module that analyzes prediction uncertainty and consistency across training, and (iii) an inductive profiling module that enables scoring of previously unseen nodes integrated into the graph. Together, these components allow GNN-MultiFix to characterize node-level learning behavior and identify failure modes beyond aggregate performance metrics. We begin by introducing the notations used throughout the paper and then describe the components of GNN-MultiFix.

Notations. Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ represents the node set and $\mathcal{E}$ represents the edge set, each node $v \in \mathcal{V}$ is associated with a feature vector $\mathbf{x}_v \in \mathbb{R}^d$, where d is the feature dimension, and a label $y_v \in \{1, 2, \ldots, C\}$, where C is the total number of classes in the graph. We use $\mathcal{V}_c$ to denote all the nodes belong to class c, and $|\mathcal{V}_c|$ to indicate the size of class c. The one-hot encoded label vector of a node v is denoted as $\mathbf{y}_v \in \{0, 1\}^C$. In this work, we use the term "class" and "label" interchangeably.

5.3.1. Data-Centric Node Profiling

This component computes interpretable scores that quantify the intrinsic difficulty of a node based on graph structure and input features. Specifically, it measures how different a node's

features are from others in the same class, diversity of labels in the local neighborhood, and higher-order structural ambiguity of node. Each of these scores are detailed below.

Intra-class Feature Dissimilarity (ICFD). Intuitively, the nodes with input features that differ significantly from other nodes in the same class are harder to classify using feature-based models. These nodes lack the typical intra-class coherence and may therefore require additional contextual information (e.g., graph structure) for accurate prediction.

To quantify this, we introduce the *intra-class feature dissimilarity* score, which captures the average cosine dissimilarity of a node v's feature vector $\mathbf{x}_v$ from those of its class peers $\mathcal{V}_c$:

$$S_{f_v} = 1 - \frac{1}{|\mathcal{V}_c| - 1} \sum_{v' \in \mathcal{V}_c \setminus v} \frac{\mathbf{x}_v \cdot \mathbf{x}_{v'}}{\|\mathbf{x}_v\| \|\mathbf{x}_{v'}\|} \tag{5.1}$$

This score serves as a lightweight, model-agnostic proxy for node difficulty: a high S_{f_v} suggests that v has atypical features relative to its class, and may challenge models that rely primarily on node attributes (e.g., MLPs). Conversely, a low score implies stronger intra-class alignment, making the node easier to classify from features alone.

Neighborhood Class Divergence (NCD). We quantify the information captured in the one-hop neighborhood of a node v by comparing the distribution of labels within this neighborhood with the average distribution of labels from the local neighborhoods of other nodes belonging to the same class.

This score is inspired by the notion of cross-class neighborhood similarity proposed in [15], which extends beyond traditional homophily. As shown in [15], effective class separation in GNNs does not require nodes of the same class to be directly connected. Instead, it is sufficient for their local neighborhoods to exhibit similar structural or label characteristics. Under this condition, models like GNNs can learn to embed such nodes into similar regions of the latent space, thereby facilitating accurate label prediction.

Specifically, we measure the difference of the label distribution in the direct neighborhood of node v in class c to the typical label distribution in the direct neighborhood of nodes in class c using Kullback–Leibler divergence (KL-Divergence). Let $\mathscr{P}_v$ represent the normalized label distribution in the immediate neighborhood of node v, defined as: $\mathscr{P}_v(c) = \frac{\sum_{u \in \mathcal{N}(v)} \mathbf{y}_u}{\sum_{u \in \mathcal{N}(v)} \sum_{c=1}^{C} \mathbf{y}_{u,c}}$, where $\mathcal{N}(v)$ denotes the set of one-hop neighbors of node v, and $\mathbf{y}_{u,c}$ denotes the c-th element in $\mathbf{y}_u \in \{0,1\}^C$. Similarly, let $\mathscr{Q}_{y_v}$ denote the average normalized label distribution in the one-hop neighborhoods of all nodes in the same class as v, i.e., class c, given by: $\mathscr{Q}_{y_v} = \frac{1}{|\mathcal{N}_c|} \sum_{v \in \mathcal{V}_c} \sum_{u \in \mathcal{N}(v)} \mathbf{y}_u$, where $\mathcal{V}_c$ is the set of nodes with label c, and $\mathcal{N}_c$ is the union of one-hop neighborhoods for all nodes in $\mathcal{V}_c$.

$$S_{l_v} = \sum_{c \in \mathscr{C}} \left[\log(\mathscr{P}_v(c) + \epsilon) - \log(\mathscr{Q}_{y_v}(c)) \right] \cdot (\mathscr{P}_v(c) + \epsilon) \tag{5.2}$$

where ϵ is the smoothing factor, which takes the value of 10^{-10} in our implementation. We include a discussion of extending this formulation to multi-label graph dataset in Appendix.

Random Walk Class Divergence (RWCD). We introduce the Random Walk Class Divergence Score (RWCD) as a measure of how mixed the higher-order neighborhood of a node v is with respect to class labels. The underlying hypothesis is that nodes surrounded by a higher proportion of different-class neighbors are harder to classify due to increased label ambiguity, typically occurring near class boundaries.

We estimate the local class distribution around a node v by aggregating the labels encountered during multiple random walks. Specifically, let $\mathscr{S}$ denote the *multiset* of nodes visited across N random walks of length k starting from v. We compute the label count vector $\mathbf{d}_w \in \mathbb{R}^C$as $\mathbf{d}_w = \sum_{j \in \mathscr{S}} \mathbf{y}_j$, which represents the frequency of each class observed in the random walk neighborhood of v. Finally, we define the RWCPS score denoted by S_h as a measure of how mixed the neighborhood of node v is with respect to its true class label c as

$$S_h = 1 - \frac{\mathbf{d}_w[c]}{\sum_{c' \in \mathscr{C}} \mathbf{d}_w[c']}, \tag{5.3}$$

where $\mathbf{d}_w[c]$ denotes the number of nodes in $\mathscr{S}$ belonging to the same class as v.

Data-centric node profiling focuses solely on the properties of the dataset. For a given graph dataset $\mathscr{G}$, the profiling score of each node depends only on the dataset itself and is independent of the model's deployment.

5.3.2. Model-Centric Node Profiling

To understand how models behave on individual nodes, GNN-MultiFix incorporates a model-centric profiling component that captures predictive dynamics during training. Specifically, we assess each node's *prediction consistency* and *prediction confidence* across model checkpoints, quantifying how stable and certain the model is about its predictions over time. Following prior work [13], we use these uncertainty estimates to define a taxonomy of node difficulty, categorizing nodes as *easy*, *ambiguous*, or *hard*.

In particular, we consider a model $\mathscr{M}_{(\theta)}$ trained on a graph $\mathscr{G}$ by minimizing a supervised objective $\mathscr{L}$ over training data $\mathscr{D}_{train}$. During training, we save a series of model checkpoints $\mathscr{E} = \{e_1, e_2, \ldots, e_E\}$, each associated with a parameter configuration θ_e. At each checkpoint e, the model outputs a predicted class probability distribution over C classes for every node $v \in \mathcal{V}$. Let $\mathscr{P}(v, \theta_e) \in [0,1]^C$ denote this predicted distribution at checkpoint e, and let $\mathscr{P}_c(v, \theta_e)$ denote the predicted probability for the true class label c of node v. We define the *mean predicted probability* for node v over training as:

$$\overline{\mathscr{P}}(v) = \frac{1}{E} \sum_{e \in \mathscr{E}} \mathscr{P}_c(v, \theta_e)$$

We then compute two types of predictive uncertainty for each node: (i) **Epistemic uncertainty**, which captures the prediction variance across checkpoints, indicating model instability:

$$v_{\text{ep}}(v) = \frac{1}{E} \sum_{e \in \mathscr{E}} \left[\mathscr{P}_c(v, \theta_e) - \overline{\mathscr{P}}(v) \right]^2, \tag{5.4}$$

and (ii) **Aleatoric uncertainty**, which measures the model's confidence in its prediction at each checkpoint:

$$v_{\text{al}}(v) = \frac{1}{E} \sum_{e \in \mathscr{E}} \mathscr{P}_c(v, \theta_e)\,(1 - \mathscr{P}_c(v, \theta_e))\,. \tag{5.5}$$

These two quantities reflect complementary aspects of node-level difficulty: epistemic uncertainty signals inconsistency in learning, while aleatoric uncertainty reflects intrinsic ambiguity in the node's label given its features and local structure. Based on these quantities, we categorize each node $v \in \mathcal{V}$ into one of three classes: *Easy*, *Hard*, or *Ambiguous*. This is formalized as follows:

$$g(v, \mathscr{G}) = \begin{cases} \text{Easy} & \text{if } \overline{\mathscr{P}}(v) \geq C_{\text{up}} \wedge \\ & v_{al}(v) < P_{50}\,[v_{al}(\mathcal{V})] \\ \text{Hard} & \text{if } \overline{\mathscr{P}}(v) \leq C_{\text{low}} \wedge \\ & v_{al}(v) < P_{50}\,[v_{al}(\mathcal{V})] \\ \text{Ambiguous} & \text{otherwise} \end{cases} \tag{5.6}$$

Here, C_{up} and C_{low} are confidence thresholds, and $P_{50}[\cdot]$ denotes the empirical median of a distribution.

5.3.3. INDUCTIVE PROFILING OF UNSEEN NODES

GNN-MULTIFIX enables profiling of previously unseen nodes introduced into the existing graph $\mathscr{G} = (\mathcal{V}, \mathscr{E})$. We design the profiling of unseen nodes solely around the uncertainties of the deployed models, since the categorization of a node (hard, ambiguous, or easy) is inherently dependent on the model and the task. The data-centric node property scores from section 5.3.1 are not used, thereby preventing potential data leakage.

Given a new node v_{new} with feature vector $\mathbf{x}_{\text{new}}$ and incident edges $\mathscr{E}_{\text{new}}$, we construct an augmented graph $\mathscr{G}' = (\mathcal{V} \cup \{v_{\text{new}}\}, \mathscr{E} \cup \mathscr{E}_{\text{new}})$. Given a model $\mathscr{M}_{(\theta)}$ trained on $\mathscr{G}$ with saved checkpoints $\mathscr{E} = \{e_1, \ldots, e_E\}$ we use the final checkpoint e_E, to compute the representation of v_{new} via a forward pass on its local subgraph consisting of L-hop neighborhood for an L-layer GNN: $\hat{\mathbf{z}}_{v_{\text{new}}}^{(e_E)} = \mathscr{M}_{(\theta_{e_E})}(\mathscr{G}_{\text{sub}})$. We then perform K-nearest neighbor search in the representation space to find $\mathcal{N}_K(v_{\text{new}})$ from the training nodes. GNN-MULTIFIX predicts a difficulty category (easy, hard, or ambiguous) via majority vote over this neighborhood:

$$\hat{P}_{v_{\text{new}}} = \text{MajorityVote}\left(\{\text{Category}_u \mid u \in \mathcal{N}_K(v_{\text{new}})\}\right) \tag{5.7}$$

To evaluate profiling accuracy, we derive the ground-truth difficulty category for v_{new} by computing prediction consistency and confidence across checkpoints (Equations 5.4–5.6). For multiple new nodes $\mathcal{V}_{\text{new}}$, the categorization accuracy of GNN-MULTIFIX on the unseen nodes is then:

$$\text{Acc} = \frac{1}{|\mathcal{V}_{\text{new}}|} \sum_{v \in \mathcal{V}_{\text{new}}} \mathbb{I}(\hat{P}_v = P_v) \tag{5.8}$$

Nodes classified as *easy* tend to receive confident and consistent predictions, while *hard* or *ambiguous* nodes exhibit instability during training, indicating potentially unreliable predictions.

5.4. Research Questions and Experimental Setup

To validate and examine the broader implications of GNN-MultiFix, which has been specifically designed to provide fine-grained characterizations of model behavior at the node level (thereby addressing the central research question posed in Section Introduction), we explore the following research questions:

RQ 5. *How can* GNN-MultiFix *be used to uncover fine-grained differences in model behavior at the node level between models with similar overall accuracy?*

RQ 6. *To what extent can* GNN-MultiFix *accurately predict the reliability of model predictions on previously unseen nodes?*

RQ 7. *How effectively can* GNN-MultiFix *identify semantically inconsistent or erroneous instances in knowledge-driven graph settings?*

5.4.1. Datasets

We demonstrate GNN-MultiFix on three diverse real-world datasets: (i) **Cora** [16], a citation network where nodes are research papers with binary word features and class labels based on topic categories; (ii) **Credit** [17], a financial graph with customer nodes and attributes related to credit risk, where we use a preprocessed subset of 3000 nodes following [18]; and (iii) **BitcoinAlpha** [19], a signed, directed trust network of trading accounts, where node features are derived from trust scores and structural statistics. In addition, to demonstrate the ability of GNN-MultiFix to identify atypical patterns and semantic inconsistencies beyond what standard benchmarks reveal we construct a node classification task from the FB15k-237 knowledge graph dataset [20] as described below and further detailed in the Appendix.

Knowledge Graph S-FB. Starting from the training subset $\mathcal{G}_{train}$ of FB15k-237 [21], we create a new graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ where each node represents a triplet $U = \langle u, r, v \rangle$. Two nodes are connected if they share an entity. Textual representations are generated using the `intfloat/multilingual-e5-large` model by concatenating the descriptions of the entities and relation. We pose a binary classification task: nodes derived from valid triplets are labeled as *true* (1), and corrupted variants—constructed by replacing one entity in 1% of triplets with a random entity—are labeled as *false* (0). To further simulate label noise, we randomly flip the labels of 30% of nodes. This setup allows us to test whether GNN-MultiFix can flag semantically incorrect or mislabeled nodes as hard or ambiguous, thereby demonstrating its utility for error detection in knowledge-driven settings.

We further summarize the characteristics of the datasets in the following Table 5.1.

Models and training set up

We consider three widely used graph neural networks: Gcn [22], Gat [23], and GraphSage [24], along with a simple multilayer perceptron (Mlp) as a baseline. In the standard training setup, we used the built-in split in Cora and Credit and randomly split the nodes in BitcoinAlpha into 60% for training, 20% for validation, and 20% for testing. We denote this setting as Partial Train in the following sections. Additionally, to assess the upper bound of model performances, we adopt a second setup in which all nodes in the graph are

Dataset	#Nodes	#Edges	#Features	#Classes
CORA	2,708	5,429	1,433	7
CREDIT	3,000	28,854	13	2
BITCOINALPHA	3,783	28,248	8	2
S-FB	272,115	996,010,50	1,024	2

Table 5.1: Summary of Graph Dataset Characteristics

used simultaneously for training, validation, and testing. We denote this setting as FULL TRAIN. The training process in these two setups are combined with early-stopping with patience of 100 epochs.

	CORA	CREDIT	BITCOINALPHA
GCN	0.801/0.981	0.613/0.661	0.850/0.850
GAT	0.780/0.997	0.594/0.641	0.594/0.732
GRAPHSAGE	0.780/1.000	0.614/0.690	0.810/0.850
MLP	0.569/1.000	0.623/0.680	0.738/0.740

Table 5.2: Prediction Accuracy of the collected machine learning models in PARTIAL TRAIN/ FULL TRAIN setups. The full table with standard deviation on three random splits is summarized in Appendix.

5.5. RESULTS

We now demonstrate how GNN-MULTIFIX can be used to compare different models in the behavior at the node level **(RQ1)**, estimation of prediction reliability **(RQ2)** and detection of semantic errors **(RQ3)**.

5.5.1. DIFFERENCES IN MODEL-BEHAVIOR (RQ1)

As shown in Table 5.2, different models often attain comparable overall accuracy on the same dataset. However, this raises a critical question: do these models with matching aggregate performance also behave similarly at the node level? In this section, we leverage GNN-MULTIFIX uncover fine-grained differences in model behavior, enabling us to move beyond coarse evaluation metrics and diagnose how and on which nodes models diverge in their learning and generalization patterns.

First, we analyze the per-model, per-category accuracy across the CREDIT, CORA, and BITCOINALPHA datasets using the predicted node categories from GNN-MULTIFIX. Specifically, for each model on each dataset, we compute the per-node classification accuracy for every test node at the final checkpoint as follows:

$$M_v = \begin{cases} 1, & \text{if } \hat{y}_v = y_v, \\ 0, & \text{otherwise,} \end{cases} \quad \text{for } v \in \mathcal{V}_{\text{test}}. \tag{5.9}$$

This yields an instance-wise accuracy vector $\mathbf{M} = [M_v]_{v \in \mathcal{V}_{\text{test}}} \in \{0,1\}^{|\mathcal{V}_{\text{test}}|}$. Next, we group the test nodes according to the difficulty category predicted by GNN-MULTIFIX, resulting

in reordered accuracy vector $\mathbf{M} = \{\mathbf{M}_{\text{easy}}, \mathbf{M}_{\text{ambiguous}}, \mathbf{M}_{\text{hard}}\} \in \{0,1\}^{|\mathcal{V}_{\text{test}}|}$. We then compute the average accuracy for each category as

$$\bar{\mathbf{M}} = \left\{ \bar{M}_{\text{easy}} = \frac{\sum_{v \in \mathcal{V}_{\text{easy}}} M_v}{|\mathcal{V}_{\text{easy}}|}, \bar{M}_{\text{ambiguous}} = \frac{\sum_{v \in \mathcal{V}_{\text{ambiguous}}} M_v}{|\mathcal{V}_{\text{ambiguous}}|}, \bar{M}_{\text{hard}} = \frac{\sum_{v \in \mathcal{V}_{\text{hard}}} M_v}{|\mathcal{V}_{\text{hard}}|} \right\}. \quad (5.10)$$

The resulting per-category classification accuracies are summarized in Table 5.3. This analysis provides a deeper understanding of how each model's overall performance decomposes across difficulty levels. A trustworthy model should exhibit high confidence and consistency on the easy nodes, indicating robust generalization to unseen data. As shown in Table 5.3, for Cora and BitcoinAlpha, correct predictions are largely concentrated within the easy and ambiguous categories, with models failing to correctly classify nodes predicted as hard. In contrast, for Credit, the accuracy is more evenly distributed across categories, suggesting that the models encounter more complex and inconsistent patterns in this dataset, leading to the limited overall performance observed in Table 5.2.

Model	Cora			Credit			BitcoinAlpha		
	Hard	Ambiguous	Easy	Hard	Ambiguous	Easy	Hard	Ambiguous	Easy
Gcn	0.000	0.797	0.819	0.769	0.540	0.751	0.000	0.816	0.981
Gat	0.000	0.766	0.912	0.521	0.602	0.594	0.000	0.544	0.966
GraphSage	0.000	0.000	0.780	0.650	0.612	0.623	0.000	0.751	0.904

Table 5.3: Per-model, per-category accuracy across datasets Cora, Credit, and BitcoinAlpha using predicted categories from GNN-MultiFix.

Then, we take Credit as an example to conduct a quantitative analysis on the final saved checkpoints of the trained models in the Partial Train setup. Note that we refer to the categorization defined in Equation 5.6 as the ground-truth categorization, and to the KNN-derived categorization as the predicted categorization. We use these terms consistently throughout this paper.

Although the models exhibit similar overall performance in Table 5.2, their behaviors differ significantly. As shown in Figure 5.2, comparing the three graph machine learning models—Gcn, Gat, and GraphSage —we plot the predicted categorization made by GNN-MultiFix on the test data. Gcn treats most training samples as easy and the fewest as hard. In contrast, Gat struggles with the largest number of test samples, finding 71 of them hard to classify. This indicates that Gcn is better trained and generalizes more effectively to unseen data than Gat. When comparing Gcn with GraphSage, the main difference lies in the handling of ambiguous data: GraphSage identifies 834 out of 1,000 test nodes as ambiguous, reflecting a higher level of uncertainty in its predictions.

Following the model-centric analysis, we now take dataset Credit as an example and conduct a quantitative data-centric analysis. Specifically, we ask: Are there node properties that are consistent with obtained characterizations of easy or hard nodes across different GNN models? To investigate this, we plot the distributions of data-centric scores for training nodes that are identified as easy(in green) or hard(in red) by all evaluated graph machine learning models.

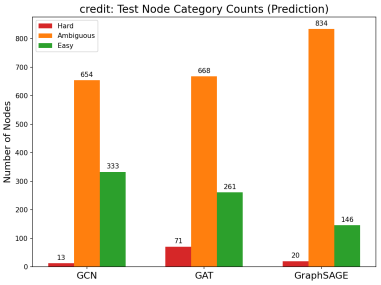
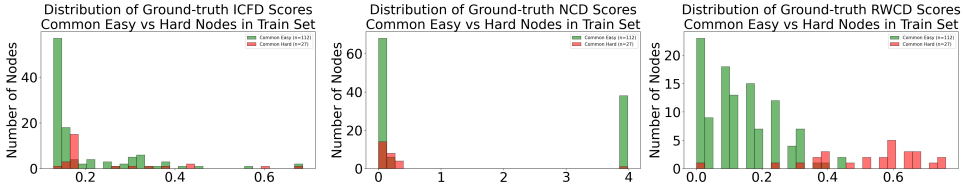


Figure 5.2: Models with similar overall accuracy show completely different certainty of their predictions on unseen data.

As shown in Figure 5.3, there are 112 common easy and 27 common hard nodes in the training set of CREDIT. Overall, the majority of easy nodes exhibit lower data-centric profiling scores (ICFD and NCD), indicating that their input features and local neighborhoods are more homogeneous relative to other nodes in the same class. Their lower RWCD scores indicate that the random-walk neighborhoods of these nodes are dominated by same-class nodes, meaning their higher-order context is less label-mixed and exhibits low class ambiguity.

By examining the distribution of hard nodes in the training set, we gain insight into why GNNs struggle to learn them correctly. In the CREDIT dataset specifically, these nodes typically exhibit larger deviations in their raw input features compared to other nodes of the same class, as indicated by their higher ICFD scores (around 0.2 in Figure 5.3a), whereas the easiest nodes tend to collapse toward an ICFD score near 0. In contrast, the NCD scores are far less discriminative, as shown in Figure 5.3b. We hypothesize that, although classical GNNs aggregate only from the immediate neighborhood at each layer, the multi-hop propagation of messages effectively exposes a node to higher-order neighbors. For hard nodes, these higher-order regions contain a greater mix of different-class labels, a pattern consistent with the distribution observed in Figure 5.3c.



(a) ICFD distributions for common easy/hard nodes.

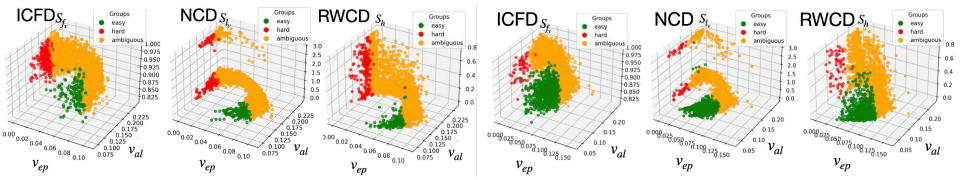
(b) NCD distributions for common easy/hard nodes.

(c) RWCD distributions for common easy/hard nodes.

Figure 5.3: Common Easy vs Hard node across GNN models have different data-centric profiling score distributions in the training data in CREDIT.

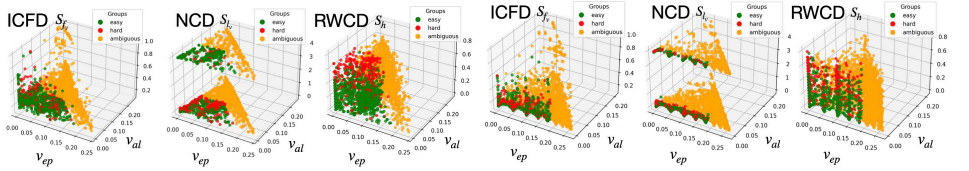
Finally, we analyze the predictions of the models on CORA, CREDIT and BITCOINALPHA datasets in correlation scatter plots (as in Figs. 5.4 to 5.8). Each point represents a node in a given graph dataset, plotted with its epistemic uncertainty (v_{ep} as x-axis), aleatoric uncertainty (v_{al} as y-axis), with z-axis depicting one of the data profiling scores. Points are colored based on their assignments by GNN-MULTIFIX for all the nodes using the uncertainties defined in equation 5.4, 5.5 and 5.6.

PARTIAL TRAIN set-up. As shown in Table 5.2, GAT and GRAPHSAGE achieve the same node classification accuracy on CORA in the PARTIAL TRAIN set-up. However, a comparison of the plots in Figures 5.4a and 5.4b reveals notable differences in their behavior. Specifically, GRAPHSAGE demonstrates higher prediction confidence, with GNN-MULTIFIX assigning



(a) GAT on CORA. **(b)** GRAPHSAGE on CORA.

Figure 5.4: GAT and GRAPHSAGE achieve similar performance on CORA but with completely different behavior on the nodes.



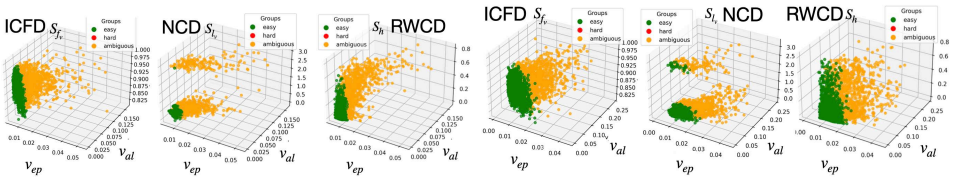
(a) GCN on CREDIT using built-in splits. **(b)** GRAPHSAGE on CREDIT using built-in splits.

Figure 5.5: GCN and GRAPHSAGE on CREDIT have comparable performance, but GCN finds more nodes with higher profiling scores easy.

1296 out of 2708 nodes to the easy category and only 57 to the hard category. In contrast, GAT exhibits lower confidence, with only 115 nodes categorized as easy and 223 as hard. This distinction is further reflected in lower values along the v_{al} axis for GRAPHSAGE, indicating more stable prediction confidence. Notably, for both GAT and GRAPHSAGE, hard nodes show extremely low v_{ep} values, suggesting that these nodes are predicted incorrectly and consistently so throughout training.

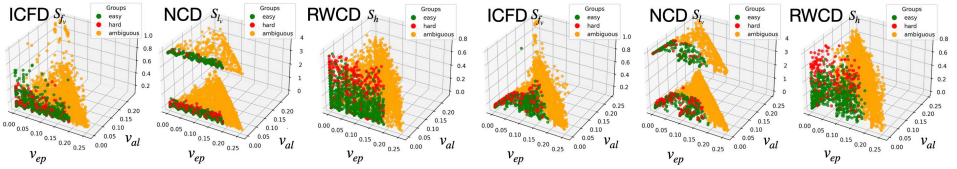
Considering the data-centric scores on the z-axis, GAT particularly struggles with nodes that have high intra-class feature dissimilarity (ICFD) and neighborhood class divergence (NCD) scores, indicating significant differences in input features and label distributions within their local neighborhoods, whereas GRAPHSAGE is able to partially learn from or generalize to such nodes, despite their challenging characteristics.

In contrast for CREDIT (Figure 5.5) one can observe larger number of easy nodes that have high neighborhood class-divergence (NCD) scores. Easy and hard nodes are better separated for GCN than GRAPHSAGE where hardness seem to increase for GCN with higher



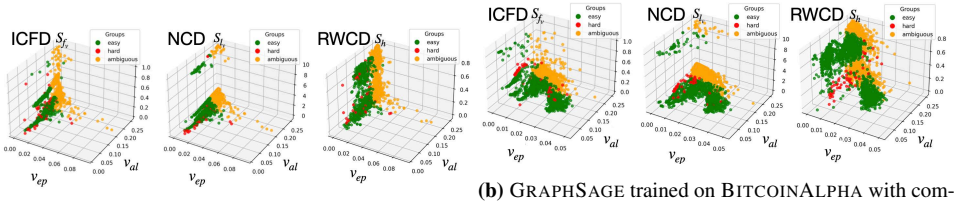
(a) GRAPHSAGE on CORA trained with complete data. **(b)** MLP on CORA trained with complete data.

Figure 5.6: Diagnosing GRAPHSAGE and MLP on CORA in FULL TRAIN setup.



(a) GRAPHSAGE trained on CREDIT with complete data. **(b)** MLP trained on CREDIT with complete data.

Figure 5.7: Diagnosing GRAPHSAGE and MLP on CREDIT in FULL TRAIN setup



(a) GCN trained on BITCOINALPHA with complete data.

(b) GRAPHSAGE trained on BITCOINALPHA with complete data.

Figure 5.8: Diagnosing GCN and GRAPHSAGE on BITCOINALPHA in FULL TRAIN setup

values of random walk class divergence (RWCD) scores.

FULL TRAIN set-up. As summarized in Table 5.2, the FULL TRAIN setup represents an upper-limit scenario in which all nodes in the graph are simultaneously used for training, validation, and testing.

In Figure 5.6, we compare GRAPHSAGE and MLP, both achieving perfect accuracy on CORA in the FULL TRAIN setup. For GRAPHSAGE, model-centric scores are compact and well-separated: easy nodes cluster tightly at the bottom-left, while ambiguous nodes form a broader but coherent region. In contrast, MLP (Figure 5.6b) shows more dispersed clusters, with ambiguous nodes exhibiting higher variance in v_{al} and several reaching v_{ep} values up to 0.25, suggesting overfitting to easy nodes and less stable predictions on ambiguous ones. With abundant training data, MLP exhibits greater confidence in its correct predictions on nodes with high neighborhood-class divergence compared to GRAPHSAGE. One might then ask: *is leveraging explicit graph structure always beneficial when sufficient training data is available?*

The answer to the above question might not be trivial when we observe results on **CREDIT** dataset (Figure 5.7). Despite using the complete data for training none of the two models could not make correct predictions on all nodes. For both the best performing models (GRAPHSAGE and MLP) there is no clear separation among easy and hard nodes on any of three axes.

In contrast to CREDIT, we see a better separation of easy and hard nodes in BITCOINALPHA (see Figure 5.8) dataset from GRAPHSAGE along the z-axis depicting random walk class divergence (RWCD) score. This is not the case for the other best performing model i.e. GCN. When considering model-centric scores GCN shows compact and stable uncertainty estimates, with v_{ep} mostly in the range $[0.00, 0.04]$ and v_{al} between $[0.10, 0.25]$. Easy nodes

form a dense cluster with low v_{ep} and moderate v_{al}, hard nodes show only a slight increase in v_{ep}, and ambiguous nodes remain tightly grouped near-zero in v_{ep} and close to the upper bound of v_{al}. Despite this concentration easy and hard nodes cannot be easily separated for GCN making it harder to analyze the reasons for the observed hardness. More detailed analysis of all remaining cases can be found in the Appendix.

5.5.2. Predicting Reliability in Predictions (RQ2)

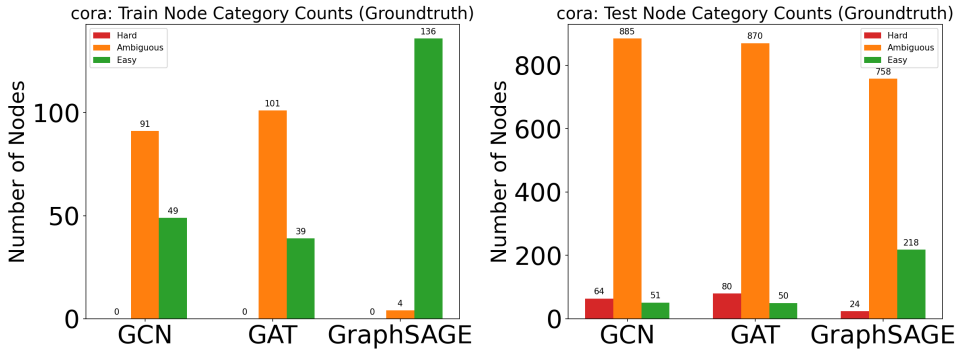
We use GNN-MultiFix to predict the difficulty category *easy*, *ambiguous*, or *hard* for previously unseen (test) nodes by applying a majority vote over their K-nearest neighbors (from the training set) in the model's output representation space. This is done via a single forward pass over the subgraphs of the test nodes using the best-performing model checkpoint from the Partial Train setup. We fix $K = 10$ for all experiments. Table 5.4 reports the categorization accuracy across three datasets—Cora, Credit, and BitcoinAlpha—using four models: GCN, GAT, GraphSAGE, and MLP. GNN-MultiFix consistently demonstrates high accuracy in assigning difficulty labels to unseen nodes. Importantly, nodes predicted to be in the *easy* category are typically associated with confident and stable predictions, suggesting that their model outputs can be trusted.

Interestingly, although GraphSAGE achieves good overall performance on Cora, GNN-MultiFix shows low categorization accuracy. We investigate this by visualizing the distribution of ground-truth categories on the training nodes, as well as the predictions and ground-truth categorization on the test nodes, in Figure 5.9.

As shown in Figure 5.9a, GraphSAGE exhibits significantly higher confidence on the training data compared to GCN and GAT, despite their similar overall performance. Specifically, GraphSAGE treats the majority of training nodes as *easy*, while GCN and GAT view most training nodes as *ambiguous*. None of the models consider any training nodes to be *hard* to classify. This indicates that, with the built-in split, the models quickly and easily fit the 140 training nodes.

In Figure 5.9b, we present the ground-truth categorization of the test nodes in Cora. We observe that GraphSAGE identifies a larger number of nodes as easy relative to GCN and GAT. In contrast, GCN and GAT produce distributions for the test nodes that more closely mirror those of the training nodes: the majority of nodes are categorized as ambiguous, with a smaller portion labeled as easy. GraphSAGE, however, exhibits a stronger mismatch: while most training nodes are labeled as easy, most test nodes are categorized as ambiguous. This contrast suggests that GraphSAGE may be overconfident on the training set, likely due to overfitting to the small number of training nodes.

When considering the prediction categorization reported in Table 5.4, GNN-MultiFix reflects this mismatch by yielding low categorization accuracy for GraphSAGE on the Cora dataset. Since the majority of the training nodes are labeled as easy and only four are ambiguous, a test node is, under the KNN clustering algorithm, most likely to be predicted as easy and only rarely as ambiguous, while it is impossible for a test node to be predicted as hard. This serves as an indicator that, although the model appears confident on the training nodes, this confidence does not align with its actual behavior on unseen nodes.



(a) Train nodes ground-truth categorization counts on dataset CORA

(b) Test nodes ground-truth categorization counts on dataset CORA

Figure 5.9: Comparison of distribution of node categorization on dataset CORA.

	CORA	CREDIT	BITCOINALPHA
GCN	0.773	0.662	0.945
GAT	0.830	0.601	0.982
GRAPHSAGE	0.218	0.657	0.703
MLP	0.092	0.672	0.958

Table 5.4: Categorization accuracy of test nodes achieved by GNN-MULTIFIX using only train nodes.

5.5.3. IDENTIFYING SEMANTIC ERRORS (RQ3)

We train GRAPHSAGE on our constructed graph S-FB with one random split to observe the training dynamics on all the nodes, which includes both valid and intentionally corrupted (false) nodes (triples). As the model is exposed to predominantly correct data, we hypothesize that model-centric profiles from GNN-MULTIFIX can flag semantic inconsistencies, i.e., nodes whose learned patterns contradict established knowledge. Due to the scale of S-FB, we train with a batch size of 1024. GNN-MULTIFIX classifies 34.9% of test nodes as easy, 50% as ambiguous, and 15.1% as hard.

Crucially, GNN-MULTIFIX effectively identify corrupted nodes. As the corrupted nodes can be randomly splitted into train-, val-, and test-set. We use the ground-truth categorization for analysis. Overall, 97.3% of the hard nodes correspond to flipped-label nodes, which encode contradictions to ground-truth facts. Overall, 48.9% of all flipped-label nodes are categorized as hard, demonstrating GNN-MULTIFIX's ability to surface potential errors through model behavior profiling.

Further to qualitatively demonstrate the capability of GNN-MULTIFIX in identifying erroneous entries or deviations from typical knowledge in the constructed graph, we conducted a manual inspection of hard nodes flagged by the GNN-MULTIFIX. For instance, hard node #272114 contains the corrupted triple $U = \langle$*Paul Williams*, nominated for, *Bad Boy Records*$\rangle$, which was originally $U = \langle$*Paul Williams*, nominated for, *A Star Is Born*$\rangle$. Similarly, another hard node #201387 includes $U = \langle$*Velvet Goldmine*, film/music, *Carter Burwell*$\rangle$, which was

incorrectly labeled as *False* after label flipping. Both cases correspond to injected semantic errors and highlight the potential of GNN-MULTIFIX to surface such anomalies for further scrutiny.

To further investigate the behavior of GNN-MULTIFIX, we analyze the neighborhood of a specific hard node (index 201387) in the S-FB graph. We randomly select six of its neighbors—two from each predicted difficulty category: *easy*, *ambiguous*, and *hard*. Table 5.5 summarizes the semantic content of these neighbors, along with whether their labels were flipped during dataset construction.

As shown in Table 5.5, all selected neighbors correspond to semantically valid triples, as indicated by their original (unflipped) labels. However, nodes with flipped labels are more frequently categorized as *ambiguous* or *hard*. These nodes are connected to the target node via shared entities like *Velvet Goldmine* and *Carter Burwell*, suggesting that GNN-MULTIFIX can identify semantic inconsistencies based on neighborhood patterns and learned difficulty profiles.

Category	Entity1	Relation	Entity2	Label Flipped
easy	Howl	Film/Music	Carter Burwell	No
easy	True Grit	Film/Music	Carter Burwell	No
ambiguous	bisexuality	Netflix genre	Velvet Goldmine	Yes
ambiguous	Eddie Izzard	Film	Velvet Goldmine	No
hard	film score	Music genre	Carter Burwell	Yes
hard	Velvet Goldmine	film location	London	Yes

Table 5.5: Example nodes from S-FB with difficulty category, semantic content and whether an incorrect label was used during training.

5.6. CONCLUSION

In this work, we introduced GNN-MULTIFIX, a node profiling framework that captures fine-grained differences in graph model behavior by assigning interpretable scores to individual nodes. These scores combine data- and model-centric perspectives to identify failure patterns that aggregate metrics like accuracy or loss often miss. We demonstrated that the profiles generalize to unseen nodes, enabling prediction reliability assessment without ground-truth labels, and effectively detect injected errors in a knowledge graph. While our framework demonstrates strong effectiveness, a limitation lies in the current scope of data-centric profiles, which may not fully capture finer-grained structural and semantic variations across nodes. As future work, we plan to broaden these profiles and explore their application to tasks such as model selection, as well as extend the framework to dynamic and multi-label graph settings.

5.7. APPENDIX / SUPPLEMENTAL MATERIAL

This appendix provides additional technical details, extended results, and supplementary discussions that support the analyses in the main paper.

In Section *Discussion on Multi-Label Datasets*, we extend our formulation of the neighborhood class divergence scorre to the multi-label node classification setting, complementing the single-label formulation presented in Section *Our Proposed Framework* of the main paper. Section *Model Hyperparameter Setting and Infrastructures* gives detailed descriptions of the datasets used and the hyperparameter settings for all models, supporting the experimental setup introduced in the main paper. In Section *Additional Experimental Results*, we present further experimental results including node categorization performance across multiple random splits of the BITCOINALPHA dataset, expanding the results presented for predicting reliability in predictions (RQ2). Section *Correlation Analysis Between Data- and Model-based Node Profiling Scores* provides detailed analyses of model-centric and data-centric profiling scores for different models and datasets. We then include additional insights on model behavior during training in Section *Training Behavior Analysis*, which compares the behavior differences of the models shown on different categories of the given datasets. Finally, in Section S-FB *Without Flipping Labels*, we present an ablation study on the necessity of label flipping in the S-FB dataset.

5.7.1. DISCUSSION ON MULTI-LABEL DATASETS

As mentioned in the main paper, in the problem formulation, we focus on the multi-class node classification in this work, where one node can only belong to one class. To include the possibility of multi-label node classification [25], where one node can belong to more than one class, the formulation of S_{l_v} can be summarized as the average difference between $\mathscr{P}_v$ and averaged label distribution in the direct neighborhood of each class $y_v \in \mathscr{Y}_v$:

$$S_{l_v} = \frac{1}{|\mathscr{Y}_v|} \sum_{y_v \in \mathscr{Y}_v} \sum_{c \in \mathscr{C}} \Big[\log(\mathscr{P}_v(c) + \epsilon) - \log(\mathscr{Q}_{y_v}(c)) \Big] \cdot (\mathscr{P}_v(c) + \epsilon) \tag{5.11}$$

where we report the average NCD to every class node v belong to. Higher value of S_{l_v} indicates a very different label distribution in the intermediate neighborhood of node v compare to other nodes of the same class.

DATASETS

CORA [16] is a citation network in which each node represents a research article, and an edge connects two nodes if one article cites the other. Each node is labeled with the category of the corresponding article. Node features are binary word vectors (0/1), indicating whether a specific word appears in the abstract of the article.

CREDIT [17] contains customer-level financial data used to predict credit default risk. Attributes include demographics, credit history, and loan information, with some versions incorporating graph structures to represent user relationships. It supports tasks like credit scoring and risk assessment. We followed the pre-processing process in [18] to extract 3000 nodes out of the original CREDIT dataset.

BitcoinAlpha [19] is a signed, directed network comprising trading accounts, where each node corresponds to an individual account and each edge represents a trust rating issued by one user toward another. Edge weights range from −10 (indicating complete distrust) to 10 (indicating complete trust). Following the procedure described in [18], we construct node feature vectors based on received trust ratings and structural properties of the network, such as in-degree and out-degree.

S-FB is a knowledge graph constructed from the original FB15K-237 dataset [21]. Each node in the graph corresponds to a triple of the form $<u, e, v>$. Edges between nodes are established based on shared entities, i.e., an edge exists if two nodes share at least one common entity. Node labels indicate whether the corresponding node contains randomly replaced entity within the triple. nodes containing such errors are assigned to class 0, while error-free nodes are assigned to class 1. Furthermore, 30% of the nodes have intentionally flipped labels, which introduces noise into the classification. We put the empirical evidence for the necessity of flipping the labels in the Appendix S-FB Without Flipping Labels.

5.7.2. Model Hyperparameter Setting and Infrastructures

In this section, we summarize the hyperparameter configurations employed in this study for the collected machine learning models (shown in Table 5.6) as well as for GNN-MultiFix.

For all publicly-available models we retained the default settings provided in their respective GitHub repositories, because our objective is not to benchmark absolute performance but to probe the learning dynamics of each model on the target dataset. As noted in Section Training Behavior Analysis, we additionally sampled training 80 checkpoints with uniform increment to minimize any effects from differences in training length.

Dataset	Model	Hidden Dim	LR	Num Layers
Cora	GCN	64	0.01	2
Cora	GAT	64	0.01	2
Cora	MLP	64	0.01	2
Cora	GraphSAGE	64	0.01	2
BitcoinAlpha	GCN	256	0.001	2
BitcoinAlpha	GAT	256	0.001	2
BitcoinAlpha	MLP	256	0.001	2
BitcoinAlpha	GraphSAGE	256	0.001	2
Credit	GCN	256	0.001	2
Credit	GAT	256	0.001	2
Credit	MLP	256	0.001	2
Credit	GraphSAGE	256	0.001	2
S-FB	GraphSAGE	256	0.001	2

Table 5.6: Hyperparameter Settings. "LR" indicates learning rate.

For the choice of thresholds c_{up} and c_{low} in GNN-MultiFix, we used 0.75 and 0.25 for the datasets Cora, Credit, and BitcoinAlpha. For S-FB, we used $c_{up} = 0.6$, $c_{low} = 0.4$.

All experiments were conducted on high-performance computing servers featuring 2 × Intel® Xeon® Gold 6140 CPUs @ 2.30GHz and 2 × AMD EPYC 7452 32-Core Processors,

providing a combined infrastructure with a total of 72 Intel threads and 128 AMD threads.

5.7.3. ADDITIONAL EXPERIMENTAL RESULTS

We built three random splits on BITCOINALPHA and report the performance of the four collected models in the PARTIAL TRAIN setup in the following Table 5.7:

	Classification Accuracy	Profiling Accuracy
GCN	0.850 ± 0.003	0.878 ± 0.080
GAT	0.590 ± 0.005	0.942 ± 0.068
GRAPHSAGE	0.835 ± 0.020	0.777 ± 0.093
MLP	0.743 ± 0.004	0.965 ± 0.011

Table 5.7: Average node classification accuracy achieved by GCN, GAT, GRAPHSAGE, and MLP on BITCOINALPHA, and the node categorization accuracy achieved by GNN-MULTIFIX with standard deviation.

Take the best performing model on BITCOINALPHA in average GCN as an example, we summarize the test categorization in Table 5.8:

Metric	**split 0**	**split 1**	**split 2**
Total test nodes	758	758	758
Easy	138 (18.21%)	176 (23.22%)	109 (14.38%)
Ambiguous	625 (81.13%)	572 (75.46%)	641 (84.56%)
Hard	5 (0.66%)	10 (1.32%)	8 (1.06%)

Table 5.8: Test node categorization in BITCOINALPHA across three random splits using GNN-MULTIFIX.

5.7.4. CORRELATION ANALYSIS BETWEEN DATA- AND MODEL- BASED NODE PROFILING SCORES

We analyze the predictions on the three collected graph datasets in $3D$ correlation scatter plots in details in this section. The plots follow the same layout as in the main paper, where each point represents a node in a given graph dataset, plotted with its epistemic uncertainty (v_{ep} as x-axis), aleatoric uncertainty (v_{al} as y-axis), and one of the profiling scores (S_{f_v}, S_{l_v}, or S_h as z-axis). Points are colored by group difficulty (easy in green, ambiguous in orange, or hard in red) as assigned by GNN-MULTIFIX using the uncertainties defined in equation 5.4, 5.5 and 5.6.

PARTIAL TRAIN

CORA As shown in Figure 5.10, GCN exhibit a clear stratification of easy, ambiguous, and hard groups. Among all the nodes in the graph, easy samples identified by GNN-MULTIFIX are concentrated in regions of both low epistemic (v_{ep}) and aleatoric (v_{al}) uncertainty, indicating that GCN not only predicts the correct labels but does so consistently with low

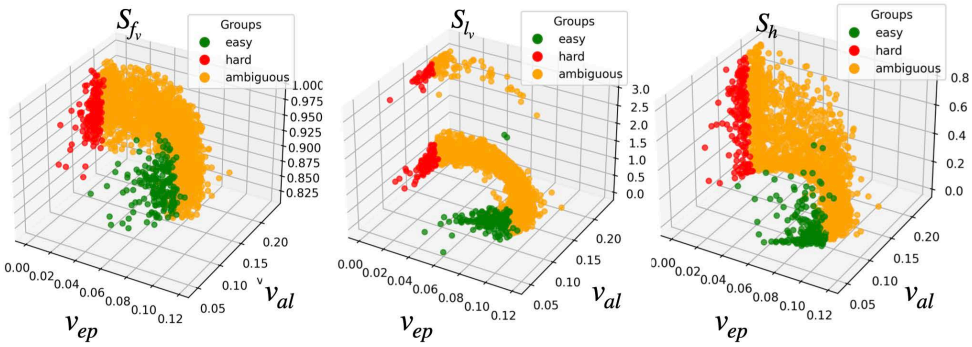


Figure 5.10: PARTIAL TRAIN correlation scatter plots of GCN trained on CORA using built-in splits.

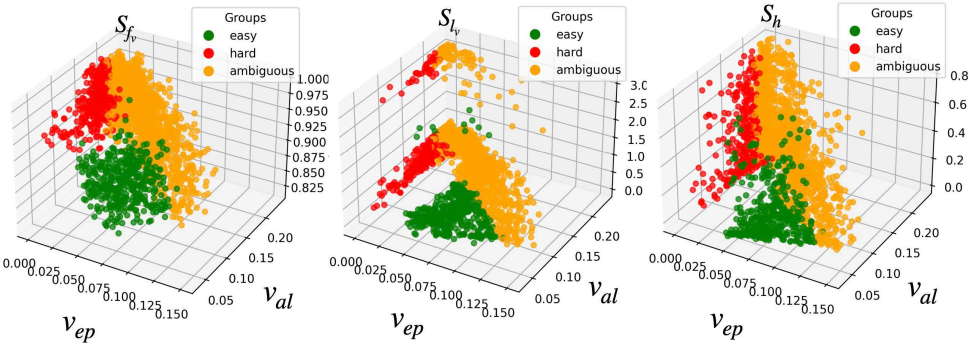


Figure 5.11: PARTIAL TRAIN correlation scatter plots of MLP trained on CORA using built-in splits.

variance. This suggests strongly confident and consistent predictions for these instances. Nodes identified as easy by GNN-MULTIFIX tend to have lower data-based node profiling scores compared to ambiguous and hard nodes. This suggests that the easy nodes are more similar (in terms of features, local neighborhoods, and higher-order structures) to other nodes of the same class. Comparing with MLP illustrated in Figure 5.11, GNN-MULTIFIX also categorizes part of the nodes for GCN with high ICFD and NCD scores as ambiguous whereas MLP solely depends on the input features and shows worse generalization for such challenging nodes.

The gap in the distribution of S_{l_v} across all collected models highlights a key characteristic of dataset CORA: the immediate neighborhood of the node in CORA tends to be either highly similar to the majority or distinctly different, with few cases where nodes partially share neighborhoods with others from the same class.

CREDIT Figures 5.12 and 5.13 present the correlation scatter plots for GAT and MLP evaluated on the dataset CREDIT under the PARTIAL TRAIN setting. Interestingly, while the MLP classifier achieves the highest test accuracy in this setup (as reported in Table 5.2), it concurrently demonstrates the lowest consistency or confidence in its predictions on the

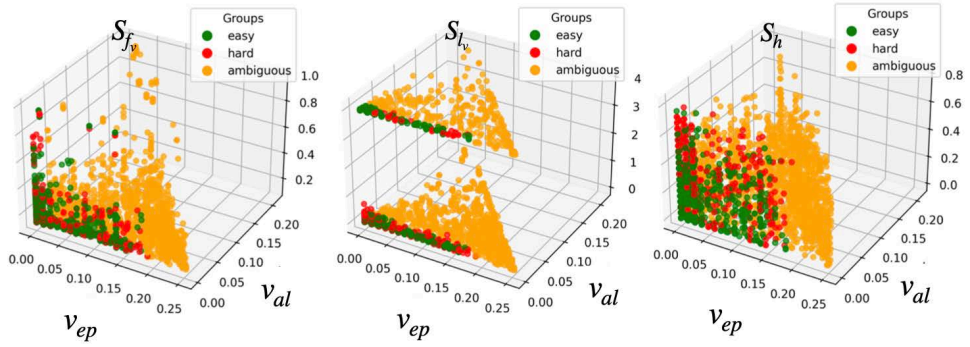


Figure 5.12: PARTIAL TRAIN correlation scatter plots of GAT trained on CREDIT using built-in splits.

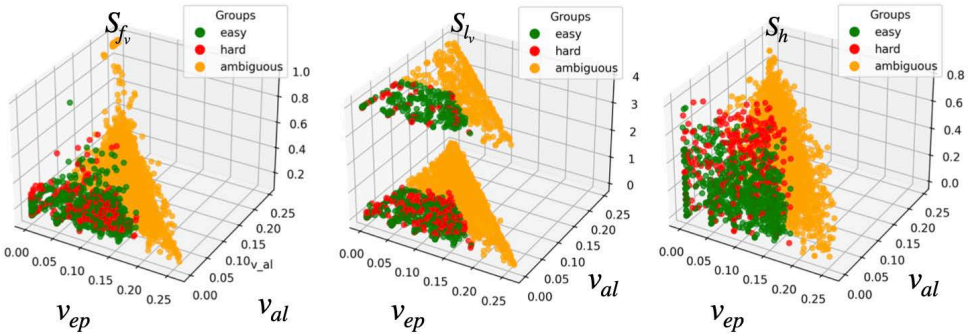


Figure 5.13: PARTIAL TRAIN correlation scatter plots of MLP trained on CREDIT using built-in splits.

unseen test nodes (as indicated by the categorization accuracy in Table 5.4). This suggests MLP may overfit to the easy nodes in the train and test subsets and struggles to generalize to the ambiguous/hard nodes in the test subset. This observation further highlights that high test accuracy alone may not fully reflect the quality of the learned representations, and that evaluation frameworks like GNN-MULTIFIX, which reflects the confidence and consistency of the predictions are crucial for a more comprehensive assessment of model reliability.

BITCOINALPHA Figures 5.14 to 5.17 present the correlation scatter plots for the collected models on the BITCOINALPHA dataset. Compared to the other two datasets, nodes categorized as easy, ambiguous, and hard by GNN-MULTIFIX tend to exhibit higher v_{al} values and lower v_{ep} values when using GCN, GAT, and MLP. This pattern suggests that model confidence on BITCOINALPHA is generally lower than on other datasets. Even when predictions are correct, the predicted probability for the true label is not substantially higher than for the alternative class.

In contrast, the scatter plots for GRAPHSAGE exhibit a noticeably different distribution. When examined alongside data-centric node profiling, we observe that many nodes categorized as easy by GNN-MULTIFIX tend to have low ICFD and NCD values. These

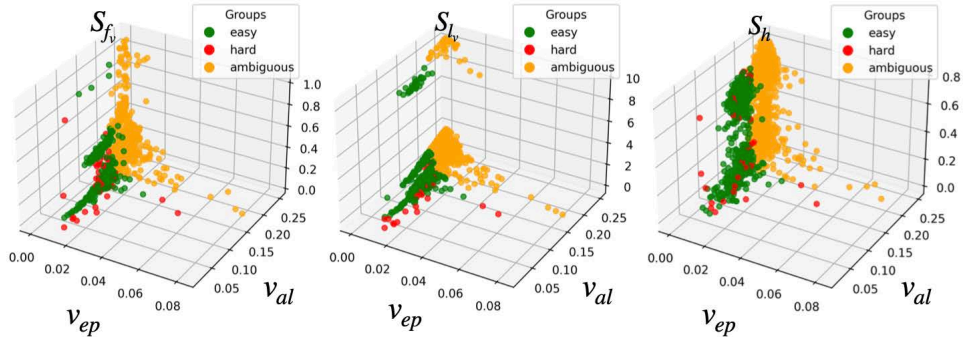


Figure 5.14: PARTIAL TRAIN correlation scatter plots of GCN trained on BITCOINALPHA.

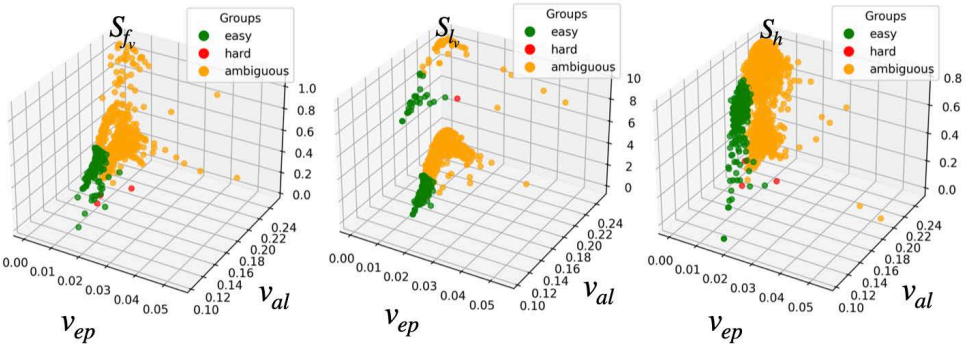


Figure 5.15: PARTIAL TRAIN correlation scatter plots of GAT trained on BITCOINALPHA.

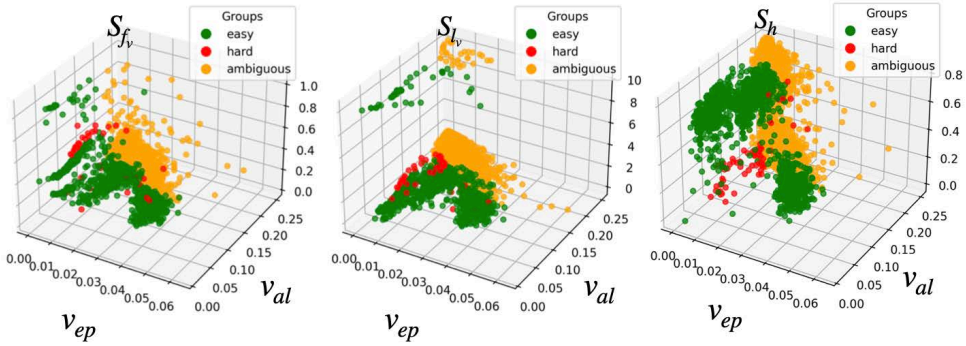


Figure 5.16: PARTIAL TRAIN correlation scatter plots of GRAPHSAGE trained on BITCOINALPHA.

characteristics suggest that such nodes are structurally and semantically more aligned with the rest of the nodes in their class. In comparison, the same nodes are often categorized as ambiguous when evaluated with other models. Additionally, GNN-MULTIFIX assigns only a small fraction of nodes to the hard group for GRAPHSAGE, reflecting a more consistent

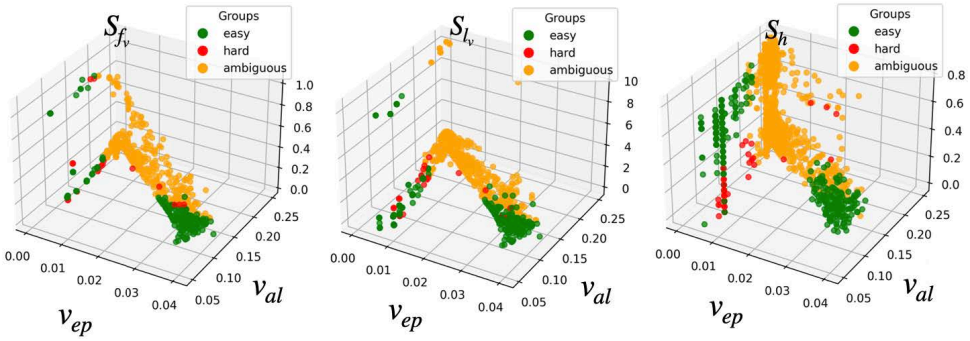


Figure 5.17: PARTIAL TRAIN correlation scatter plots of MLP trained on BITCOINALPHA.

and confident prediction pattern. These observations indicate that the node subsets identified as easy by GNN-MULTIFIX exhibit properties that GRAPHSAGE can generalize to more effectively, underscoring its adaptability to the structural nuances of the BITCOINALPHA dataset.

FULL TRAIN

CORA As shown in Figure 5.18, the GCN model trained under the FULL TRAIN setting exhibits improved performance compared to the PARTIAL TRAIN setting (Figure 5.10). Specifically, a larger number of nodes with high ICFD and RWCD scores are categorized as easy by GNN-MULTIFIX. Additionally, nodes categorized as easy in PARTIAL TRAIN with low NCD are predicted with lower variance in FULL TRAIN, as reflected in reduced v_{ep} values, demonstrating substantially lower variance in prediction. However, the ambiguous nodes (orange) appear more scattered than the ambiguous nodes in the PARTIAL TRAIN setup, though they remain relatively distinct from the easy group. This dispersion suggests that under FULL TRAIN, GCN may overfit to the easy instances, while still struggling to generalize to more uncertain, ambiguous cases. As a result, GCN may rely heavily on memorized patterns from confidently learned nodes, leading to reduced robustness when encountering structurally or semantically complex inputs.

As shown in Figure 5.19, GAT exhibits a similar trend to GCN in terms of data-centric node profiling. However, the ambiguous nodes (orange) in GAT are more widely spread along the v_{ep} axis compared to GCN. This suggests that although GAT achieves low uncertainty for easy nodes (green), its attention mechanism may inadvertently introduce noise or overemphasize less informative neighbors, resulting in less stable representations for ambiguous cases. Nevertheless, the distinction between easy and ambiguous node groups remains largely preserved.

CREDIT Figures 5.20 and 5.21 present the results of the correlation analysis conducted on the CREDIT dataset using GCN and GAT under the FULL TRAIN setup. Compared to the PARTIAL TRAIN setting, GNN-MULTIFIX categorizes more nodes with high ICFD, NCD, and RWCD scores as easy for both GCN and GAT, with comparably similar ranges of

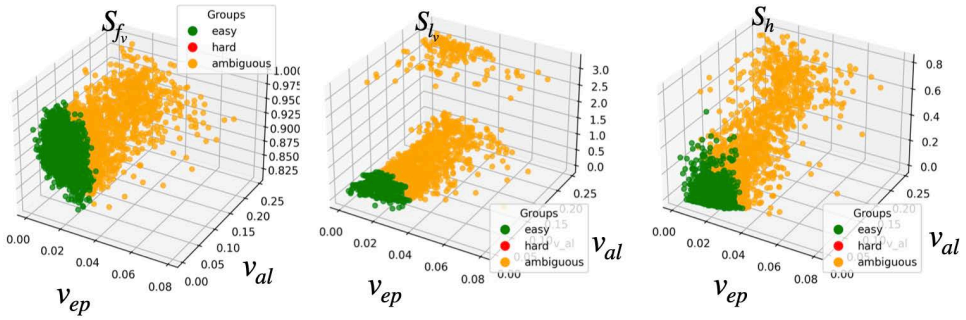


Figure 5.18: FULL TRAIN correlation scatter plots of GCN trained on CORA.

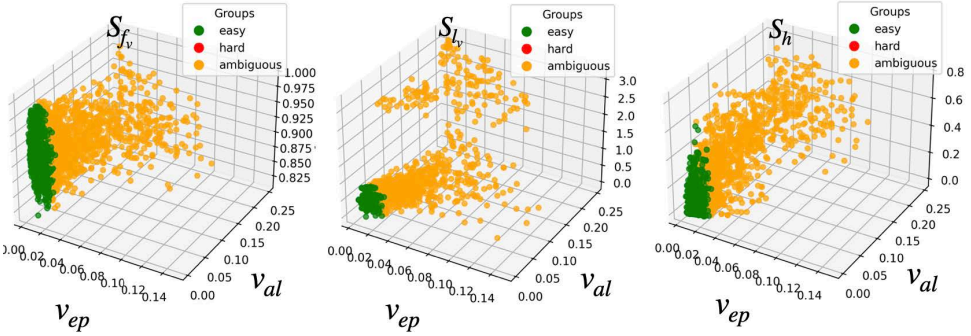


Figure 5.19: FULL TRAIN correlation scatter plots of GAT trained on CORA.

uncertainties. This observation suggests that, when provided with sufficient training data, GCN and GAT generalize better to the challenging nodes with stable predictions.

BITCOINALPHA Figures 5.22 and 5.23 show the 3D correlation scatter plots for the BITCOINALPHA dataset under the FULL TRAIN setting using GAT and MLP. According to GNN-MULTIFIX's categorization, more nodes are labeled as easy and fewer as hard for GAT, and the overall spread of the ambiguous group becomes tighter along the uncertainty axis compared to GAT in PARTIAL TRAIN. This indicates that GAT leverages the additional supervision to refine its predictions.

In contrast, MLP, while having access to structural statistics encoded in the input features, such as degree and edge weight distributions, does not explicitly model the relational graph structure. As a result, it exhibits only subtle changes in both uncertainty measures under the FULL TRAIN setting. There is a slight reduction in v_{ep} and a modest tightening of the ambiguous cluster in the ICFD and RWCD plots. However, the separation between nodes categorized as easy and hard remains weak.

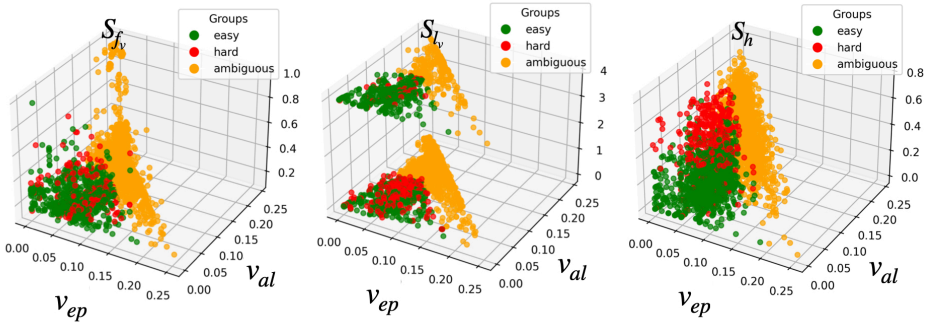


Figure 5.20: FULL TRAIN correlation scatter plots of GCN trained on CREDIT.

5.7.5. TRAINING BEHAVIOR ANALYSIS

In this section, we analyze the behavior of the collected models in the PARTIAL TRAIN setup with respect to nodes categorized as easy, ambiguous, and hard—represented in green, orange, and red, respectively. Specifically, we track and summarize the average training loss for each node category across sampled checkpoints, visualizing the results using line plots. In all plots, the x-axis denotes the indices of the sampled training checkpoints, while the y-axis represents the corresponding average training loss. These subgroups were determined based on the ground-truth categorization of all nodes defined in equation 5.4, 5.5 and 5.6, and the probabilities were computed via forward passes on the graph datasets without backpropagation.

CORA As shown in Figure 5.24, across all models, the easy group shows a consistent and rapid increase in predicted probability, quickly converging to high confidence in the correct class. The MLP, despite lacking neighborhood aggregation, still achieves near-perfect accuracy on these easy nodes, indicating that their correct classification is largely feature-driven.

In contrast, the hard group consistently receives very low predicted probabilities for the true class, regardless of the model architecture. For the GCN, GRAPHSAGE, and MLP in particular, the probability for hard nodes stagnates early or even declines slightly during training. This suggests that these nodes are either poorly represented in the feature space or

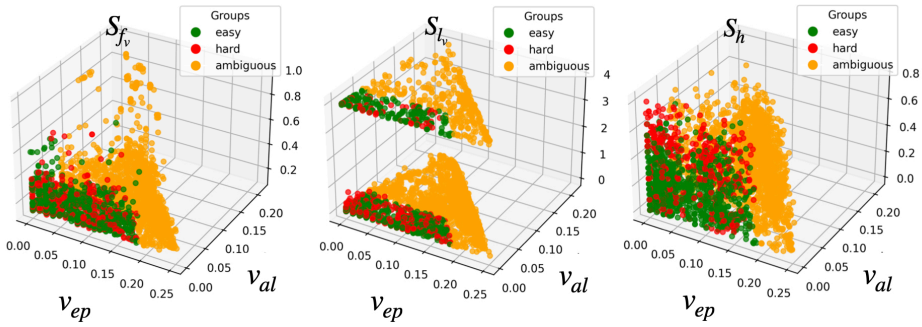


Figure 5.21: FULL TRAIN correlation scatter plots of GAT trained on CREDIT.

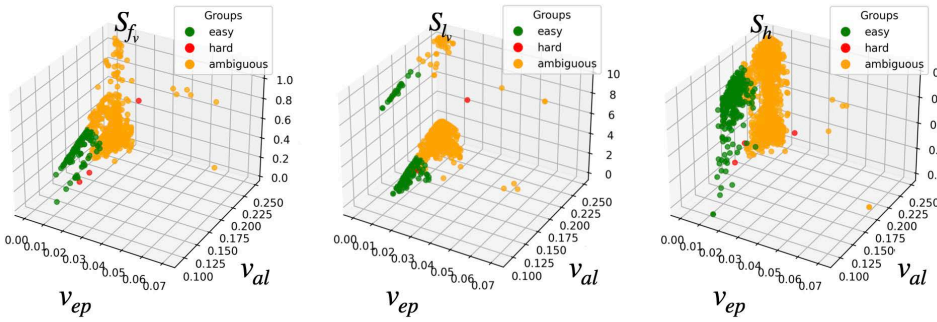


Figure 5.22: FULL TRAIN correlation scatter plots of GAT trained on BITCOINALPHA.

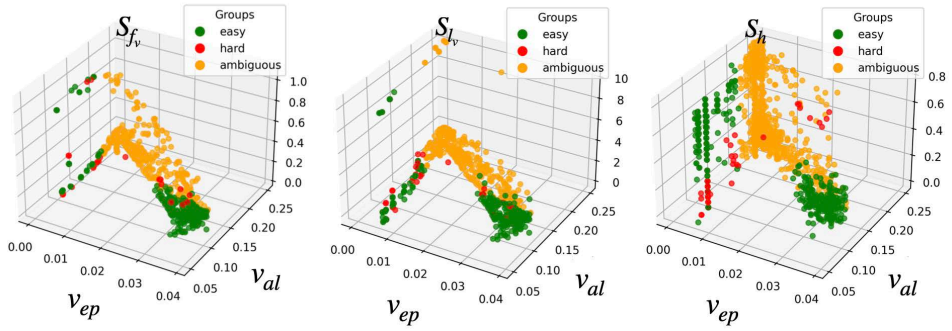


Figure 5.23: FULL TRAIN correlation scatter plots of MLP trained on BITCOINALPHA.

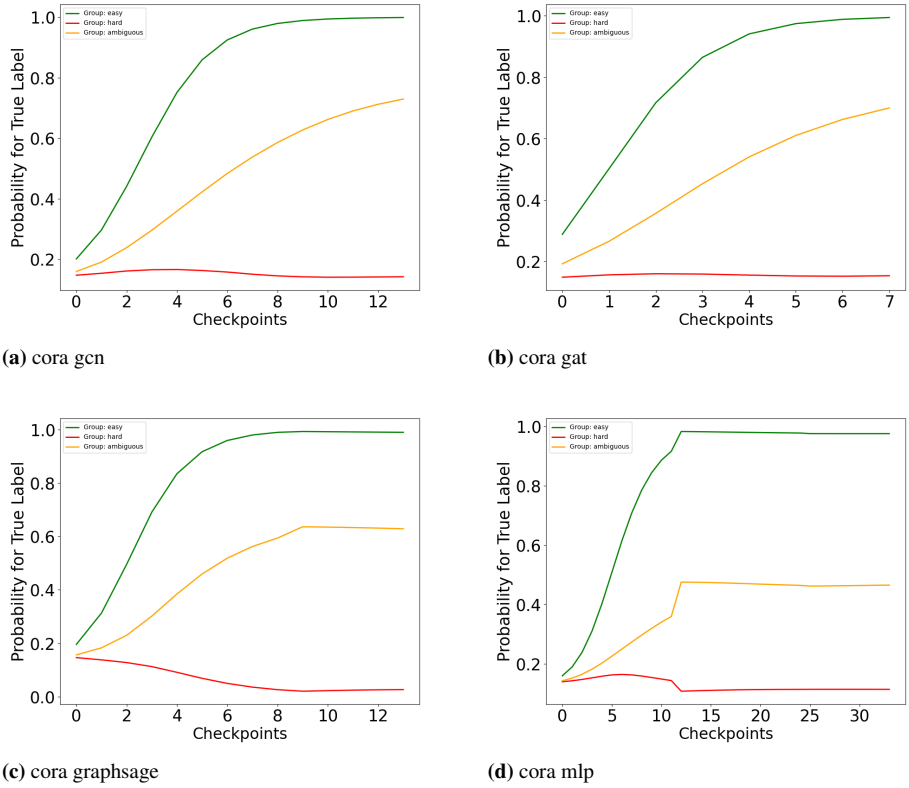
misaligned with their neighborhoods, making them persistently difficult to classify.

The ambiguous group occupies an intermediate position in terms of learning dynamics. The rate and extent of this improvement vary by architecture: GCN and GAT show relatively stronger performance, while, GRAPHSAGE and MLP show weaker progression. This indicates that the structures of GCN and GAT are more effective at incorporating structural cues to disambiguate challenging examples. In contrast, GRAPHSAGE exhibits only modest gains, suggesting limited flexibility in adapting to uncertain or noisy neighborhoods. MLP, which lacks access to graph structure entirely, demonstrates the slowest improvement, relying solely on node features and showing limited capacity to resolve ambiguity through training alone. These differences underscore the importance of relational inductive bias in improving performance on less clearly defined instances, and position the ambiguous group as a sensitive indicator of a model's capacity for representation refinement and generalization.

Overall, the training dynamics reveal that while all models are highly effective at fitting easy nodes, they struggle with hard nodes. The ambiguous group provides an informative gradient of difficulty, and the degree to which its performance improves reflects each model's capacity for generalization. These observations motivate further exploration of specialized training strategies or architectures to better handle ambiguous and hard examples in graph-structured data.

CREDIT As shown in Figure 5.25, despite the clear separation between groups, the curves on CREDIT exhibit greater fluctuation compared to other datasets, indicating a less stable training process.

The easy group consistently reaches high predicted probabilities across all models, typically plateauing above 0.9. This suggests that the features associated with these nodes are highly discriminative, enabling models, even those lacking relational structure such as



(a) cora gcn

(b) cora gat

(c) cora graphsage

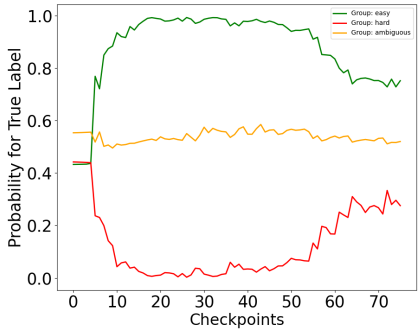
(d) cora mlp

Figure 5.24: Observed average accuracy across categories at different model checkpoints on dataset CORA.

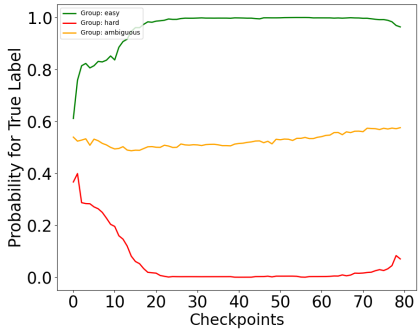
MLP, to quickly gain confidence in their predictions.

Interestingly, during the early stages of training, as the confidence of the model in the easy nodes increases, the predicted probabilities for the hard nodes tend to decrease. This pattern reflects an early overfitting tendency, where the model prioritizes easily learnable patterns at the expense of more challenging examples. In the later stages of training, particularly for GCN and GAT, the predicted probabilities for the hard group begin to rise, however, this improvement comes with a trade-off, as the confidence in the easy group starts to decline. This shift suggests a redistribution of model focus that may reflect capacity limitations or conflicting gradients across subgroups.

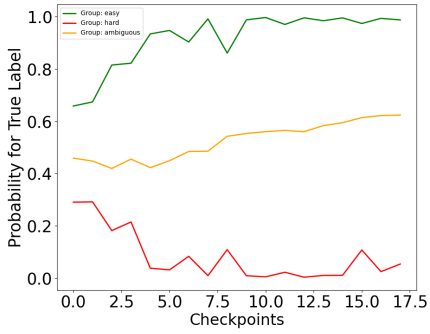
The ambiguous group occupies a stable middle ground. For GCN, GAT, and MLP the predicted probability for this group remains relatively flat. GRAPHSAGE exhibits a similar trajectory but with slightly more upward movement.



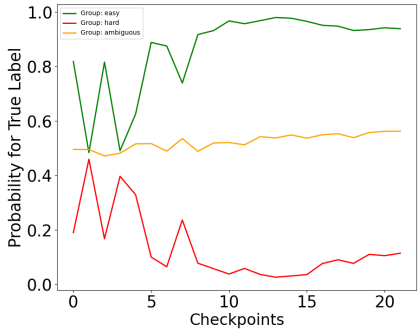
(a) credit gcn



(b) credit gat



(c) credit graphsage



(d) credit mlp

Figure 5.25: Observed average accuracy across categories at different model checkpoints on dataset CREDIT.

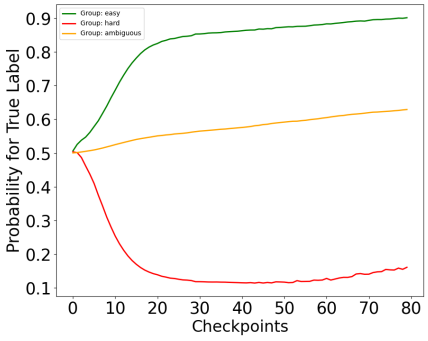
BITCOINALPHA Overall, the categorization of nodes into easy, ambiguous, and hard demonstrates the consistent ability of GNN-MULTIFIX to distinguish between different characteristics of the nodes. As shown in Figure 5.26, all evaluated models exhibit high confidence and low variance on the easy nodes. In contrast, ambiguous nodes display moderate and gradual improvement, while hard nodes consistently show low confidence and greater variance throughout training.

Both GCN and GRAPHSAGE rapidly increase their confidence on easy nodes during the early training stages, reaching probabilities near 0.9, which then remain stable until the end of training. In comparison, ambiguous nodes improve more slowly, converging around 0.6 by the final epochs. Interestingly, during the early training phase, when confidence on easy nodes is rising, the confidence on hard nodes actually declines from approximately 0.5 to 0.1. Notably, GCN shows a slight late-stage recovery for hard nodes after the 60-th checkpoint.

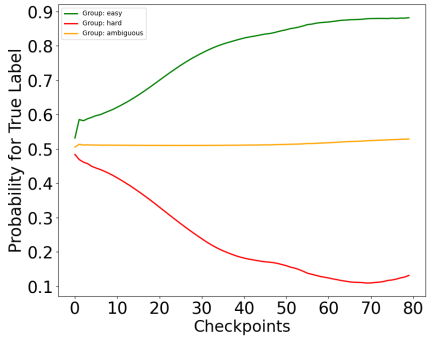
GAT follows a similar trend to GCN in its treatment of node subgroups. Easy nodes are learned quickly, reaching around 0.88 in probability, while ambiguous nodes exhibit minimal gains over training and converge near their initialization level of 0.51. Among the

models considered, GAT demonstrates the least improvement on ambiguous nodes. For hard nodes, the predicted probability for the true label starts near 0.5 due to random initialization but steadily drops to approximately 0.1 as the model focuses on optimizing performance for easier examples.

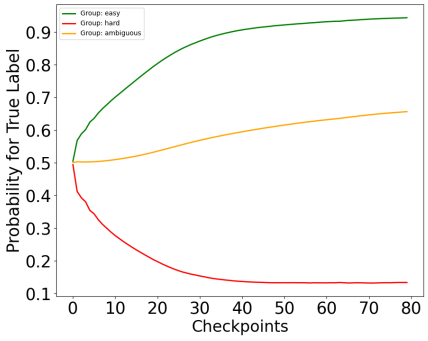
Interestingly, MLP performs comparably to other GNN models, albeit with slightly lower confidence on the easy nodes. This is likely because topological information, such as in-degree, out-degree, and edge weights, is incorporated as part of the node input features.



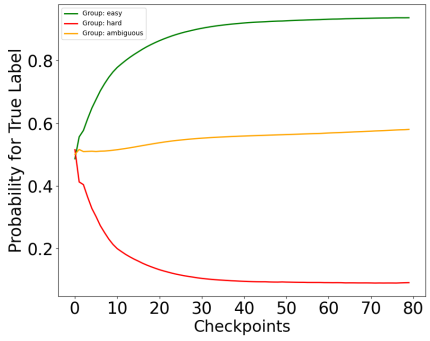
(a) bitcoin gcn



(b) bitcoin gat



(c) bitcoin graphsage



(d) bitcoin mlp

Figure 5.26: Observed average accuracy across categories at different model checkpoints on dataset BITCOINALPHA one random split.

5.7.6. S-FB Without Flipping Labels

We also investigate the categorization of nodes in the S-FB dataset without applying label flipping. In this setting, the nodes containing randomly replaced entities with label 0 effectively form a separate class that the GRAPHSAGE model attempts to fit. Table 5.9 presents the categorization statistics generated by GNN-MULTIFIX when training GRAPHSAGE on this modified dataset, which we refer to as FB15-K-Random.

	easy	ambiguous	hard
	0.493	0.507	0.00

Table 5.9: Categorization percentage assigned by GNN-MULTIFIX when training GRAPHSAGE on the dataset S-FB with 1% randomly chosen nodes injected with random replaced entities.

As shown above, all nodes were categorized into either the easy or ambiguous groups, with none falling into the hard category. Additionally, we observe that the training accuracy is nearly stationary and begins at over 0.9, which further justifies the need to inject flipped labels to introduce learning difficulty.

References

1. Zheng, X. *et al.* Towards Data-centric Graph Machine Learning: Review and Outlook. *arXiv preprint arXiv:2309.10979* (2023).
2. Jin, W. Deep Learning on Graphs: A Data-Centric Exploration. *Proceedings of the AAAI Conference on Artificial Intelligence* **38,** 22671–22671. https://ojs.aaai.org/index.php/AAAI/article/view/30287 (Mar. 2024).
3. Sun, H. *et al. Towards Data-centric Machine Learning on Directed Graphs: a Survey* 2024. arXiv: 2412.01849 [cs.LG]. https://arxiv.org/abs/2412.01849.
4. Guo, Y. *et al.* Data-centric graph learning: A survey. *arXiv preprint arXiv:2310.04987* (2023).
5. Huang, J., Shen, J., Shi, X. & Zhu, X. *On Which Nodes Does GCN Fail? Enhancing GCN From the Node Perspective* in *Forty-first International Conference on Machine Learning* (2024). https://openreview.net/forum?id=dcwUGaK9sQ.
6. Guo, Y. *et al.* Data-centric graph learning: A survey. *IEEE Transactions on Big Data* (2024).
7. Ma, X. *et al. Graph Anomaly Detection with Few Labels: A Data-Centric Approach* in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, Barcelona, Spain, 2024), 2153–2164. ISBN: 979-8-4007-0490-1. https://doi.org/10.1145/3637528.3671929.
8. Gavrilev, D. & Burnaev, E. *Anomaly Detection in Networks via Score-Based Generative Models* 2023. arXiv: 2306.15324 [cs.LG]. https://arxiv.org/abs/2306.15324.
9. Breunig, M. M., Kriegel, H.-P., Ng, R. T. & Sander, J. *LOF: identifying density-based local outliers* in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (Association for Computing Machinery, Dallas, Texas, USA, 2000), 93–104. ISBN: 1-58113-217-4. https://doi.org/10.1145/342009.335388.
10. Sehwag, V., Chiang, M. & Mittal, P. *SSD: A Unified Framework for Self-Supervised Outlier Detection* in *International Conference on Learning Representations* (2021). https://openreview.net/forum?id=v5gjXpmR8J.
11. Guo, Y. *et al. A Data-centric Framework to Endow Graph Neural Networks with Out-Of-Distribution Detection Ability* in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, Long Beach, CA, USA, 2023), 638–648. ISBN: 979-8-4007-0103-0. https://doi.org/10.1145/3580305.3599244.
12. Seedat, N., Imrie, F. & van der Schaar, M. *Dissecting Sample Hardness: A Fine-Grained Analysis of Hardness Characterization Methods for Data-Centric AI* 2024. arXiv: 2403.04551 [cs.LG]. https://arxiv.org/abs/2403.04551.
13. Seedat, N., Crabbé, J., Bica, I. & van der Schaar, M. *Data-IQ: Characterizing subgroups with heterogeneous outcomes in tabular data* 2022. arXiv: 2210.13043 [cs.LG]. https://arxiv.org/abs/2210.13043.
14. Jarrahi, M. H., Memariani, A. & Guha, S. The Principles of Data-Centric AI. *Commun. ACM* **66,** 84–92. ISSN: 0001-0782. https://doi.org/10.1145/3571724 (July 2023).

15. Ma, Y., Liu, X., Shah, N. & Tang, J. *Is Homophily a Necessity for Graph Neural Networks?* 2023. arXiv: 2106.06134 [cs.LG]. https://arxiv.org/abs/2106.06134.

16. Yang, Z., Cohen, W. W. & Salakhutdinov, R. Revisiting Semi-Supervised Learning with Graph Embeddings. *CoRR* **abs/1603.08861.** arXiv: 1603.08861. http://arxiv.org/abs/1603.08861 (2016).

17. Agarwal, C., Lakkaraju, H. & Zitnik, M. Towards a Unified Framework for Fair and Stable Graph Representation Learning. *CoRR* **abs/2102.13186.** arXiv: 2102.13186. https://arxiv.org/abs/2102.13186 (2021).

18. Olatunji, I. E., Rathee, M., Funke, T. & Khosla, M. Private Graph Extraction via Feature Explanations. *Proceedings on Privacy Enhancing Technologies* **2023,** 59–78. ISSN: 2299-0984. http://dx.doi.org/10.56553/popets-2023-0041 (Apr. 2023).

19. Kumar, S., Spezzano, F., Subrahmanian, V. S. & Faloutsos, C. *Edge Weight Prediction in Weighted Signed Networks* in *2016 IEEE 16th International Conference on Data Mining (ICDM)* (2016), 221–230.

20. Toutanova, K. & Chen, D. *Observed versus latent features for knowledge base and text inference* in *Proceedings of the 3rd workshop on continuous vector space models and their compositionality* (2015), 57–66.

21. Toutanova, K. & Chen, D. *Observed versus latent features for knowledge base and text inference* in *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality* (eds Allauzen, A., Grefenstette, E., Hermann, K. M., Larochelle, H. & Yih, S. W.-t.) (Association for Computational Linguistics, Beijing, China, July 2015), 57–66. https://aclanthology.org/W15-4007/.

22. Kipf, T. N. & Welling, M. *Semi-Supervised Classification with Graph Convolutional Networks* 2017. arXiv: 1609.02907 [cs.LG]. https://arxiv.org/abs/1609.02907.

23. Veličković, P. *et al. Graph Attention Networks* 2018. arXiv: 1710.10903 [stat.ML]. https://arxiv.org/abs/1710.10903.

24. Hamilton, W. L., Ying, R. & Leskovec, J. *Inductive Representation Learning on Large Graphs* 2018. arXiv: 1706.02216 [cs.SI]. https://arxiv.org/abs/1706.02216.

25. Zhao, T., Dong, T. N., Hanjalic, A. & Khosla, M. Multi-label Node Classification On Graph-Structured Data. *Transactions on Machine Learning Research.* ISSN: 2835-8856. https://openreview.net/forum?id=EZhkV2BjDP (2023).

6

Conclusion

6.1. SUMMARY OF CONTRIBUTIONS

In this thesis, we showed how the properties of graph data, including feature noise, label imbalance, and topology-affected label homophily, strongly shape GNN performance. Given the lack of diversity in the common benchmark graph datasets, we first introduced a tunable synthetic graph generator to produce diverse synthetic graph datasets. Building on insights derived from data analysis across both real-world and synthetic multi-label graph datasets, GNN-MULTIFIX dynamically fuses feature propagation, label propagation, and positional encoding to address noisy and overlapping labels, ultimately generating informative node embeddings for the multi-label classification task. As real-world graphs evolve over time, AGALE shifts the focus from static setting to continual learning setting and introduces a principled evaluation framework, enabling a more realistic assessment of model robustness and adaptability to evolving graph data in the multi-label setting. To further deepen our understanding of model behavior, NODEPRO integrates data-centric signals (such as feature and label uncertainty and structural ambiguity) with model-centric measures (including confidence and consistency), offering a systematic approach to diagnose node-level failure modes and distinguish model performance beyond aggregate metrics.

In Chapter 1 Introduction, we formulated four research questions. We now summarize how the contributions of this thesis work toward addressing these questions.

RQ1: To what extent do current GNNs improve on multi-label node classification tasks with more complicated homophily semantics?

This question is addressed in Chapter 2 from both theoretical and empirical perspectives. **First**, the notion of homophily requires a careful redefinition in the multi-label setting. The standard definition used in single-label classification, where connected nodes either share a label or not, is no longer applicable, because in the multi-label case nodes typically share only a subset of their labels. To capture this, we introduce a new metric that quantifies label homophily as the average Jaccard similarity between the label sets of all connected node pairs. This provides a universal, edge-level measurement of homophily that can be aggregated over the entire graph.

Second, our exploratory data analysis reveals that existing benchmark graph datasets lack diversity in terms of label distributions and, consequently, in homophily patterns observed in real-world scenarios. Using our new definition of label homophily, we propose a synthetic graph generator to produce graphs with varying homophily levels and different levels of input feature quality, the latter controlled by injected noise. These synthetic graph datasets expose clear differences in the behavior of graph neural networks (GNNs) under controlled conditions.

Finally, we systematically evaluate methods from three categories, feature-based models such as MLPs, topology-based models such as DEEPWALK, and message-passing GNNs that rely on both. Using four evaluation metrics, we analyze their performance on commonly used datasets and highlight the overly optimistic nature of AUROC in the presence of class imbalance in the multi-label graph datasets. We recommend using Average Precision instead. Our analysis further reveals that the limited performance of GNNs stems from often noisy information within node neighborhoods: ego nodes typically share only a fraction of their labels with each neighbor, making the information about the remaining neighbor labels noise in accurate label prediction of the ego nodes. This observation motivated the development of

a more effective approach for generating node embeddings in multi-label node classification, which we present in Chapter 3 in the answer of *RQ2*.

RQ2: Does integrating positional encoding and label propagation into message-passing GNNs consistently benefit multi-label prediction on graph-structured data?

We address this question in Chapter 3, motivated by empirical evidence obtained during exploratory data analysis, which highlights why positional encoding and label propagation can be beneficial for multi-label node classification.

First, we conduct an empirical study of the training dynamics of existing GNN models on real-world multi-label datasets and show that they exhibit limited learning capability in multi-label settings, even when abundant training data is available. Rather than examining only the average training loss across all nodes, we analyze the loss at the instance level, revealing that many nodes fail to be learned effectively. This provides direct, instance-level evidence supporting the limitations identified in *RQ1*.

Second, prior work on graph isomorphism has focused mostly on graph-level discrimination, while isomorphism-related failures also arise at the node level. In realistic networks, nodes that occupy the same structural role can appear far apart in the graph being part of totally different communities, yet still have indistinguishable local neighborhoods under standard GNN message passing. As a result, existing GNNs often assign identical embeddings to such nodes, failing to distinguish them despite their different global positions.

Third, we find that GNNs do not effectively exploit the label distribution of neighboring nodes in multi-label datasets. Notably, a simple baseline that predicts a node's labels by aggregating the known training labels of its neighbors consistently outperforms existing GNN architectures by a substantial margin.

Motivated by these observations, we propose GNN-MultiFix, a simple yet effective framework that leverages all available information for each node—its input features, the label information propagated from labeled neighbors, and its structural position encoded through positional encodings. Through theoretical analysis and extensive experiments, we demonstrate that GNN-MultiFix achieves substantial improvements across four real-world multi-label datasets, as well as two sets of synthetic datasets with varying input feature quality and label homophily.

Encouraged by the improvements observed in our empirical experiments, and recognizing that real-world graphs are often generated in a streaming manner, we shift our focus from the static setting to the continual learning setting in *RQ3*.

RQ3: Under which scenarios do current graph continual learning evaluation frameworks fail, and how can they be generalized for both multi- and single-label scenarios?

This research question is addressed in Chapter 4. **First**, we examine existing continual graph learning evaluation frameworks and their underlying design assumptions. We find that these frameworks are primarily developed for single-label scenarios. Although multi-label settings are occasionally mentioned, they are treated only as conceptually applicable to node-level multi-label tasks and are evaluated exclusively on graph-level datasets. Since multi-label graphs are still independent from each other, thus differ fundamentally from multi-label nodes in one graph, we confirm that the complexities of node-level multi-label continual learning have been largely overlooked.

To study this setting, we begin by directly applying existing evaluation frameworks to multi-label datasets. Through this process, we identify several critical limitations, including the lack of graph-aware data partitioning strategies, overlaps between training, validation, and test sets across tasks, and reliance on predefined class orders. These issues highlight the inadequacy of current frameworks for evaluating continual learning on multi-label node classification tasks.

Second, motivated by the realistic evolution of multi-label graphs, we define two generalized incremental learning settings and provide theoretical analysis to characterize the generated subgraphs.

Third, we refine the graph data partitioning process. Existing approaches partition graphs using generic data-splitting algorithms, which do not account for multi-label dependencies, which directly makes one node belong to multiple tasks simultaneously. We therefore introduce graph-aware partitioning methods that generate subgraphs for each time step while also producing train/validation/test splits within each subgraph avoiding the possible data leakage problem.

Finally, we categorize the classes of graph machine learning models suitable for deployment in the continual graph learning setting and evaluate state-of-the-art methods under our proposed framework to assess current progress in this area.

Building on our work in developing the evaluation framework in assessing graph machine learning models behaviors, and motivated by the substantially different node-level behaviors identified in addressing *RQ2*, we next investigate the behavior of graph machine learning models at the instance level. This naturally leads to the final research question of this thesis.

RQ4: How can we understand the correlation between model behavior and graph data properties through instance-level profiling?

This research question has been studied in Chapter 5. Prior work in data-centric graph machine learning has predominantly focused on improving the quality, structure, or manipulation of graph data itself, while model-centric approaches aim to develop effective architectures evaluated primarily through aggregate performance metrics. However, what has been largely missing in the literature is a systematic analysis that connects model behavior with the underlying properties of graph data, allowing us to understand *why* models succeed or fail at the node level.

First, we introduce three interpretable scores assigned to individual nodes. These scores capture key data properties such as Intra-class Feature Dissimilarity, Neighborhood Class Divergence, and Random Walk Class Divergence. Building on existing notions of model uncertainty, we categorize nodes into *easy*, *ambiguous*, and *hard* instances. By jointly considering data properties and model predictions, we reveal which kinds of nodes a given model finds easy, ambiguous, or difficult to fit. This provides an analytical tool, complemented by visualization capabilities, that deepens our understanding of both dataset characteristics and model behavior.

Secondly, we show that the proposed profiling signals generalize to unseen nodes, enabling reliability assessment even in the absence of ground-truth labels. Furthermore, the framework effectively detects injected errors in a knowledge graph, demonstrating its utility for practical data quality auditing and instance-level diagnostic tasks.

6.2. LIMITATIONS AND FUTURE WORK

Although our results mark a step forward, several limitations remain, paving the way for future research.

Firstly, the diversity and realism of graph datasets are still limited. All proposed methods in this thesis, while theoretically and empirically validated, have primarily been tested on real-world benchmark datasets with similar properties and on our multi-label synthetic graph datasets generated under controlled conditions at relatively small scale. This leaves an open research gap in the curation of more diverse real-world graph datasets, particularly those characterized by the properties discussed in this thesis, such as different levels of feature quality, label homophily, and label distribution. A systematic comparison of these real-world datasets with the synthetically generated graphs is also largely missing. Future research should explore the development of synthetic graph generator models that more accurately reflect the complex properties inherent in real-world multi-label graphs in both static and continual learning setting.

Additionally, there remains substantial room for improving the performance of current graph machine learning models in the multi-label node classification task in both static and continual learning settings. While GNN-MULTIFIX achieves state-of-the-art performance on the multi-label node classification task at the time of this thesis, it could benefit from further theoretical analysis of the performance bounds achievable by such fusion-based methods. Second, because GNN-MULTIFIX integrates a label propagation module and employs a positional encoding module DEEPWALK, the current implementation is limited to the semi-supervised (transductive) setting. An important direction for future work is to develop inductive variants of GNN-MULTIFIX that can operate on previously unseen nodes or subgraphs without requiring access to the entire graph during inference. Third, a valuable next step would also be to develop an explainable module for positional encoding to better understand its contribution to model performance. Last but not least, the competitive performance of simple baselines such as DEEPWALK and MAJORITYVOTE should not be overlooked. Conducting theoretical analyses to understand the success of such simple methods could provide deeper insights into the properties of natural real-world multi-label graphs and help in designing more effective approaches for this task.

Thirdly, there remains a clear lack of research on the natural evolution of multi-label graph processes. While AGALE provides the first evaluation framework for continual graph learning specifically designed for both sing-label and multi-label settings, the proposed data-partitioning algorithms remain theoretically defined and have not yet been compared against the actual evolution of real-world multi-label graphs. Examples include the evolution of social networks around individuals over time, sequences of protein-protein interaction networks that expand as new protein functions are discovered, or the evolving linkage structures of Wikipedia pages over the years. These time-stamped multi-label datasets could serve as valuable ground truth for developing and testing data-partition algorithms and evaluation frameworks like AGALE. Future work should study the properties of such temporal graphs to understand how the distributions of input features, labels, and topology change over time. For example, we observed that our data partitioning algorithms tended to make the graphs at each time step more homophilic than the original full graph. As a result, the simplest baseline without any specialized continual learning mechanism achieved strong average performance. Further analysis is needed to determine whether these similar

properties hold in real-world temporal multi-label graph datasets.

Finally, there remains significant potential to deepen instance-level profiling in graph machine learning. While NODEPRO provides the first data- and model-centric analyses of node properties and model behaviors during training, its data-centric profiles may not fully capture fine-grained structural or semantic variations across nodes. This gap is likely to become more pronounced as graph machine learning models begin to incorporate increasingly complex and nuanced data properties. Therefore, it would be valuable to design new data-centric scores that capture the properties GNNs directly rely upon, while remaining interpretable and explainable to practitioners. In addition, it would be valuable to explore the use of NODEPRO for tasks such as model selection and instance-level data diagnosis, ideally in close collaboration with domain experts. Last but not least, extending the framework to dynamic and multi-label graph settings also represents an important direction for future work.

7

ACKNOWLEDGMENTS

This thesis is far from a solo achievement. Throughout this journey, there were many moments when I doubted whether I would reach the finish line at all. That it now exists is thanks to the support of many people around me, and for this I am deeply and sincerely grateful.

First, I would like to thank my promotor, Prof. Dr. Alan Hanjalic. Your feedback has always been direct and clear, and I have greatly valued the fresh perspective you brought. You often pointed out weaknesses I had overlooked and challenged assumptions I had taken for granted. Our conversations pushed me to think more critically and to reflect on my work at a more conceptual, meta level. I learned a great deal by treating your comments as supervision signals to train and improve myself.

Gratitude also goes to my daily supervisor, Dr. Megha Khosla, for being an approachable guide throughout this PhD. Thank you for the time you invested in discussing ideas in depth and reading drafts carefully. Your efforts in training me as a researcher and your enthusiasm for research, have shaped the way I think and work, lessons that will stay with me throughout my career.

I would also like to thank Prof. Dr. Geert-Jan Houben, Prof. Dr. Ibo van de Poel, Prof. Dr. Michael Cochez, Prof. Dr. Jie Yin, and Dr. Shujian Yu for serving on the committee and for taking the time to review this thesis.

Warm thanks also go to Saskia and Kim. Having your support in complicated situations was deeply reassuring. I have learned so much from your experience and advice.

Michael and Daniel, thank you for hosting me during my visit to Vrije Universiteit Amsterdam. I learned a great deal about the research you are doing, and that visit led to a rewarding collaboration. Daniel, it has been a pleasure working with you, you have been a great mentor.

Russa, I had a great time collaborating with you on the last project of my PhD. It's always fun discussing ideas as well as experiment details with you, and I'm very happy that our collaboration led to a research paper. Thank you for inviting me to visit your lab, that was the greatest recognition I could have imagined. I wish you all the best in your career!

Being part of the friendly and supportive Multimedia Computing group at TU Delft has been a huge privilege. We celebrate each other's successes and stand by one another in difficult times. Marijn, thank you for being there for me in difficult times. Holding your hands was a great comfort for me. I wish you all the best. Andrea, Dimme, Ting, Mohamed, Vimal, Yuanyuan, Bishwadeep, Imara, Huy Son, Daniele, Antony, Li, Maosheng, Xiamei, Chengen, Bart, and Tianrui, the many research talks, MMChillings, and coffee chats with you were not only fun but also a daily release of stress. Thank you all.

Shatha, Alessia, Yidi, and my cousin friends, it has been a blessing to have you with me through all this time. Shatha, thank you for putting in the effort to host so many gatherings. The gym sessions have brought about a great change in both my body and my mind. With you all, I got to try so many different cuisines from all over the world, memories I will cherish forever.

Elise, this building felt more like a home to me because you were there. Thank you for your patience and understanding. Angelika, you have been a steady source of support to me all these years. Thank you sincerely for everything. With you, Thomas, Niels, and Johanna, Hannover always feels like home to me. Shiyu, Johanna, and Hengrui, thank you for being my pen pals in this increasingly digitalized world. Taking the time to write a proper letter feels very precious. Yuqian and Xiaofang, though separated by great distance, I remain deeply grateful for your support.

To Fenghua, Zhihao, Shilun, Yingyue, Mengyi, and Yifei, thank you for the museum visits, the group trips, the gatherings, and for reminding me that great food brings great people together. Lijun, thank you for reassuring me when I had doubts and for your kind hospitality in Edinburgh. Yumeng, thank you sincerely for all our conversations. Your sharp observations have often helped me see things more clearly.

To my parents, Wei and my family, I don't know how to fully express my gratitude to you. Without your love, support, and belief in me, I could never have come this far. You make me realize that no matter where I am, you are always with me, and knowing you're there makes me feel like I can do anything, thank you.